KULICKE and SOFFA

PCC

Kulicke and Soffa Industries, Inc., 507 Prudential Road, Horsham, PA 19044 (215) 674-2800

PCC

Peripheral Communication Controller

SOFTWARE FUNCTIONAL/DESIGN SPECIFICATION

PCCOS Module Version 1.0

SECS Module Version 1.0

Tom Kane/Factory Automation
14 Sep 84

1.0 SYSTEM FUNCTIONAL OVERVIEW

The Peripheral Communication Controller (PCC) is intended to serve as the communications interface between all K&S automatic machines and a Factory Automation host. It provides the common link in communicating from a host computer system to all K&S machines.

The PCC contains a 6809 microprocessor, 16K of on-board RAM, 44K of on-board ROM, a parallel port and two RS232 ports. The system is interrupt driven to provide priority attention to requests for communication services from a machine. Because of it's intimate knowledge of the machine to which it is connected, the PCC can gather information and package it in a SECS formatted message for transmission to the host. This places the burden of generating and processing messages on the PCC, leaving the machine to concentrate on its bonding.

Communication to the machine will be either thru shared RAM or thru a machine's floppy disk port. Communication with the host is defined by the SEMI SECS standard and will be implemented using RS232 interconnections.

The basic components of the PCC software are listed below.

1. MONITOR - Provides stand alone program load, dump, and single step execution of code.

2. PCCOS (PCC Operating System) - Provides timer support, dynamic memory management, and queue management functions.

3. I/O DRIVERS - Perform all communication functions with K&S machines and a Factory Automation host.

4. SECS MODULE - Performs the control and synchronization functions for the PCC based on SECS II node transaction protocol.

5. PERSONALITY MODULE - Provides the means to customize the PCC software to operate with different K&S machines.

Section 1 of this document describes the functions of the various components of the PCC software. Section 2 details the operation of each component and their interaction with other components of the system.

1.1 MONITOR

The debugger provides initialization and control functions for the PCC. These functions include stand alone program load, dump, single step debug and hardware testing.

1.2 PCC OPERATING SYSTEM

The operating system performs runtime functions such as timer support, queue management and dynamic allocation of message buffers within the PCC. Operating system support routines are invoked by other components of the PCC by use of Pascal functions. The actual transfer of control to the utility code is accomplished by a software interrupt (SWI2).

1.3 I/O DRIVERS

I/O drivers perform all communication functions with K&S machines and the Factory Automation system. Communication with the host system is accomplished via RS232 interconnection as defined by SECS I protocol. I/O requests are queued to drivers using a standard IO interface from Pascal. This standard interface allows software to be written without regard to the device which will perform the function. The IO request mechanism also allows a high priority request to be placed at the head of the queue, rather than at the tail after waiting requests. The IO mechanism is implemented as a SWI2 to PCCOS. Code in the operating system starts the I/O if the device is not busy, or queues the request until the driver has finished its current operation.

1.4 SECS MODULE

The SECS module performs the control and synchronization functions for the PCC, based on SECS protocol. SECS I defines the physical interconnection and the data link layer of the protocol. This layer includes line contention, master/slave relationship between host and machine, and the header and checksum to accompany the message data. SECS II defines, by means of general categories (streams) which contain detailed messages (functions), the methods of conveying information between processing equipment and a host computer system. The actual data transferred is structured into Items and Lists, allowing the data of a message to describe itself. SECS II also defines the contents of the header fields and the layout of the message in the data area (consult the SEMI SECS specification for details).

1.5 PERSONALITY MODULE

The personality module contains information necessary to handle communication with a particular K&S machine. This component knows which SECS messages a particular machine can respond to, and knows the method of communication to conduct with the machine to get the requested data. The PCC will respond to the host with a stream 9 message (unrecognized stream type) for any message it does not understand. This eliminates the need for machines to be burdened with messages that they cannot process.

The personality module will contain one Pascal procedure to handle all functions within a particular stream. These routines will be called by the SECS module by using an entry point table that resides in the low address range of the personality module. Appendix F details the layout of this entry point table which must exist in all personality modules.

Utilities needed by the stream modules while processing a message will be invoked by a SWI3 interrupt. This allows the commonly used utilities needed by all stream processors to be located within the common SECS module. Since calls between the SECS module and stream processing routines are never done using absolute addressing, the personality module and SECS module can be relinked separately and proms of different revisions can coexist within the PCC.

1.6 PCC - MACHINE COMMUNICATION

Handling of input and output will be achieved by interrupt driven drivers. These drivers will be used by the SECS routines to communicate with the host and the machine. A unique personality module (16K ROM) will customize the PCC software for the particular machine the PCC serves. Since each PCC type will have a unique interface to the machine it serves, the details of the I/O will be transparent to the SECS module code.

1.6.1 Shared RAM Interface

K&S machines based on the E500 bus will utilize shared RAM for communication. The shared RAM can be 128, 256, 512 or 1024 bytes long. Both the PCC and the machine main processor can map this shared RAM into its own address space at a page boundary. The area mapped into need not be the same physical address range on both sides. See Appendix D for a memory map of the shared RAM interface.

The shared RAM interface will support 3 modes of communication between the PCC and the machine:

1. BLOCK MODE - The PCC and machine exchange blocks of data formatted by SECS II protocol. The sender writes the data into an area of shared RAM dedicated to him and then signals the receiver that data is available. The signalling can be accomplished by interrupts on both sides of the interface or through semaphores maintained in the shared memory itself. The receiver would then read the

data and signal the sender to indicate he had processed the message. Control bytes located in shared RAM are used to define the status of the shared RAM data area and to direct the processing of that data. For the case of limited amounts (less than 512 bytes) of shared RAM, a block may require multiple exchanges to exchange a complete SECS formatted block. In this case, the control bytes describe the context of the partial message.

2. TALLY VARIABLE - A variable contained in shared RAM will be updated from the machine and read from the PCC side of the shared RAM interface. The PCC could then respond directly to queries from the host for certain important data items without bothering the machine for the data. Examples of tally variables are data items such as number of wires bonded, number of missing wires, number of empty magazine slots, and PRS failures. Communication via tally variables is strictly from the machine to the PCC. Tally variables are considered read only by the PCC, and are maintained by the machine.

3. COMMAND/DISCRETE EVENT MODE - The machine sends single byte commands to the PCC. Along with the command, the machine can send 4 bytes of data which are the parameters or arguments for that command. These commands would allow the PCC to handle alarms and to maintain status variables and counters within its own RAM if space for shared tally variables is not available. For example, a command would initialize a counter of devices bonded, and then send an increment command for that counter every time a new device was bonded. The PCC could then generate a SECS II message and send that to the host (if appropriate) each time a device was completed. This moves the burden of generating the message from the bonder to the PCC.

Machines under operating system control will probably queue SECS II messages to a driver to be sent to the PCC in block mode. This method is slower than the machine writing directly to the shared RAM, but provides for synchronization of message flow and allows the user program to continue processing while the communication completes.

Certain messages(such as throughput statistics) may need to be sent to the PCC often. Rather than incur the overhead of preparing a SECS message, the bonder could write a command byte to the appropriate area of shared RAM. The command byte is polled by the PCC to determine if a command from the bonder is waiting.

A typical command exchange using shared RAM is outlined below.

1. The machine reads the command byte to determine if it is available for use. If the command byte is zero, proceed to step 3.

2. If the command byte is non-zero, the PCC has not yet serviced a previous command. The machine must delay 40 msec and attempt step 1 again. NOTE: 40 msec is the maximum time that must exist between 2 consecutive

commands. Time critical applications may wish to continually check the condition of the command byte to minimize the waiting time. In this case, the machine would give up after 40 msec of waiting for the command byte to become available.

3. The machine places 4 command data bytes (if appropriate) into shared RAM, and writes a command token into the command byte. It is imperative that data be written before the command token to avoid the situation where the PCC begins servicing the command before all data has been written by the machine.

The shared RAM interface operates in a full duplex communication mode. In this sense the machine may be sending data to the PCC while at the same time a message from FAS is being received by the machine. Both the PCC and the machine have the capability to signal to the other and indicate the reason by setting semaphores in the command area of shared RAM. These condition semaphores are write-only(WO) on the side of shared RAM originating the condition, and area read-write(R/W) on the handling side. The originator sets a semaphore and then interrupts the other processor. After handling the signaled condition, the semaphore is reset. The condition semaphores are reset at powerup by the side having R/W access. Appendix D details the layout of the shared RAM address space. A transmitter and a receiver exist on each side of shared RAM. The transmitter on each side can signal a 'data ready' or 'timeout' condition. The transmitter fields the 'data received' or 'abort' conditions from the receiver on the other side of shared RAM. The receiver on each side can generate a 'data received' or 'abort' condition. The receiver also fields the 'data ready' and 'timeout' conditions from the transmitter on the other side of shared RAM.

PCC	EQUIPMENT
TRANSMITTER	RECEIVER
data ready -->	<-- data received
timeout -->	<-- abort

RECEIVER	TRANSMITTER
data received -->	<-- data ready
abort -->	<-- timeout

The meaning of condition semaphores is detailed below:

1. DATA READY (TRANSMITTER to RECEIVER) - A message is waiting in the sender's write area.

2. REPLY TIMEOUT (TRANSMITTER to RECEIVER) - The reply expected from the host (stream and function indicated in control area of shared ram) was not received within the SECS T3 time.

3. DATA RECEIVED (RECEIVER to TRANSMITTER) - The receiver has removed the data from the sender's write area, thus the sender's write area is now available for use.

4. ABORT (RECEIVER to TRANSMITTER) - The current message in the sender's write area cannot be accepted. The sender is responsible for notifying the application who initiated the transmission of the message. If the message was part of a multi-block trasmission, the personality module should cancel the entire message (ie issue CANMSG).

A typical exchange of data using shared RAM is outlined below. In the discussion below, the sender and receiver could be either the PCC or the bonder.

1. The sender desires to send a SECS message to the receiver. His first step is to determine if his write area is available for use by checking the state of the "available flag" in his control byte. Since it is set, the sender proceeds to place his message in his write area.

2. The sender signals the receiver (via "data ready" semaphore) that data is ready in his write area of shared RAM. This data will remain in the sender's write area until the receiver acknowledges it's receipt or the sender times out. Flags in the sender's control byte indicate if the data is the "last sub-block" of a message, and if the data is the "last message" of a SECS message.

3. The receiver processes the message from the sender's area, and signals (via "data received" semaphore) that he has received the data from the sender.

4. OR

5. The receiver cannot process the data and signals (via "abort" semaphore) an error to the sender.

6. If the receiver has signaled successful receipt of the data, the sender prepares to send the next sub-block or next SECS message as appropriate. If the receiver has signaled an abort, the transmitter must notify the application who initiated the transmission.

1.6.2 Parallel Port Interface

The Parallel port interface will be used to interface the PCC to the 1419, 1470 and possibly the 775 machine. Communication will be handled using the variable length protocol defined within this document. The actual transfer of data is via a 8 bit wide parallel port. The PCC cannot interrupt the machine, but must indicate that it wishes to transmit by sending a special response back to the equipment when polled.

Communication from the machine to the PCC is either a command exchange or the transfer of a block of data formatted according to SECS protocol. The command exchange and block exchange are detailed in the following sections. All exchanges of data are verified for correctness by the receiver by using a checksum value sent after the data.

The machine sends the following commands and inquiries to the PCC.

1. COMMAND - The following data represents a command.

2. MESSAGE - The following data represents a SECS message.

3. MSG INQUIRY - inquiry if the PCC has any messages to send to the machine.

The PCC responds to the above commands and inquiries using the following responses.

1. MSG WAITING - A message is waiting for the machine to read.

2. ACK - Last message or command received sucessfully.

3. NAK - Last message or command received was not valid (ie checksum indicated that data had been lost or garbled).

The machine sends the following status replies to the PCC

1. ACK - Last message received sucessfully.

2. NAK - Last message received was not valid (i.e. checksum indicated that data had been lost or garbled).

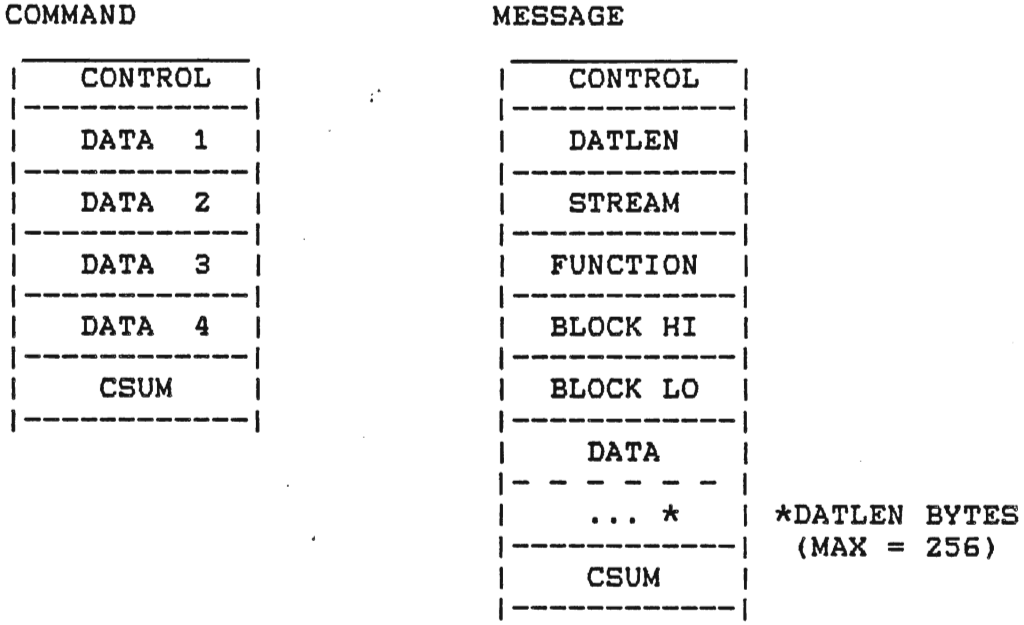
The physical communication channel consists of an 8 bit data bus, 4 control and handshake lines and 1 interrupt line (used by the bonder to interrupt the PCC). Both the bonder and PCC each have a "ready" line (which is toggled to indicate data is on the bus or has been received) and a "signal" line used to indicate that more data is to be transfered.

To send a message to the PCC, the bonder lowers it's signal line (EQPSIG), places a code on the data bus indicating it wishes to transmit a message, and interrupts the PCC by toggling the PCCINT line. The PCC responds to the interrupt by reading the control information on the data port, and then toggles it's ready line (PCCRDY) to indicate that it has seen the control information and that the bonder may safely put the first data character on the data bus. The PCC now waits for the bonder's ready line (EQPRDY) to toggle indicatating that new data is available. When EQPRDY toggles the PCC reads the data on the bus (and toggles PCCRDY), and also checks the bonder signal line (EQPSIG) to determine if more data is to be sent. If more data is coming the PCC and bonder continue to handshake in this manner.

When all data has been received (EQPSIG no longer active) the PCC calculates a checksum on the received data to determine if it was correctly received. If so it responds to the bonder with a success status, if not the PCC responds with an error status, allowing the bonder to start the transaction again at the beginning.

When the bonder is able to process requests from the PCC, it issues a "polled read" to the PCC asking if the PCC has any messages from the FAS host to send to the bonder. If the PCC responds to this inquiry indicating that a message is waiting, the bonder must go into receive mode and perform the same handshaking protocol as described above to allow the PCC to transmit.

COMMAND & MESSAGE LAYOUTS



TYPICAL EXCHANGES

1.6.2.1 Command Exchange

Command exchanges allow a command byte and 4 data bytes to be quickly sent from the machine to the PCC. The PCC responds to the command with a one byte status reply.

1.6.2.2 Message Exchange

A message exchange allows data to be sent between the machine and PCC. The data is formatted using a modified SECS format.

2.0 SOFTWARE COMPONENTS

Since the PCC is designed to be compatible with existing K&S machines as well as future products, the software is designed with flexibility and future modification in mind. To achieve this goal the PCC software is designed using a modular approach. The major software components of the PCC are discussed in the following sections.

1. Monitor/Debugger
2. PCC Operating System
3. I/O Drivers
4. SECS Module
5. Personality Module

The following sections of this document detail the design and operation of each component of the PCC and their interaction.

2.1 MONITOR

The monitor provides the ability to initialize the PCC at powerup and the means to configure the PCC in such a way to allow the controlled execution of code. The debugger can be entered by executing a SWI opcode which causes the 6809 to mask all interrupts, enter the debugger, dump all registers and prompt the debugging terminal for a command. The continue command or the single step option restarts processing at the point it was interrupted, however real time interrupts that may have occured could be lost.

2.1.1 Download

2.1.1.1 General Information

a) Function Category : Monitor
b) Function Name : DOWNLOAD
c) Activation : L command
d) Response : ACK or NAK

2.1.1.2 Purpose

The download function receives assembled code in Mikbug format from a host system and loads it to PCC RAM. This function is used only for development purposes.

2.1.1.3 Function Input

ASCII data (MOTOROLA format) containing executable code. Each record contains the address, the length of data to follow and a checksum byte. Only S0,S1 and S9 records are accepted.

2.1.1.4 Normal Processing

After receiving the L (Load) command from the development system, the PCC responds with an ACK to indicate its readiness to receive code. The host sends the code to the PCC one record at a time. The monitor decodes the address of the data, computes a checksum and stores the data received in the appropriate memory location. After each record is checked and stored, the monitor responds by sending an ACK to the host.

2.1.1.5 Abnormal Processing

If the debugger calculates a checksum on the data just received which does not match the checksum sent by the host, a NAK is sent to the host. This will inform the host that the record was received incorrect and to retransmit the last record. The host will retransmit the record up to its maximum retry count (usually 5) before quitting.

2.1.2 Verify

2.1.2.1 General Information

a) Function Category : Monitor
b) Function Name : VERIFY
c) Activation : V command
d) Response : ACK or NAK

2.1.2.2 Purpose

This function verifies the most recent downloaded code.

2.1.2.3 Normal Processing

The ASCII based code is downloaded but instead of storing it in RAM, the debugger compares the just received data record to what is currently in RAM. If the data matches, an ACK is returned to the host.

2.1.2.4 Abnormal Processing

If the data record does not match, a NAK is sent to the host.

2.1.3 Display/Modify Memory

2.1.3.1 General Information

a) Function Category : Monitor
b) Function Name : MEMORY
c) Activation : Mnnnn-mmmm or Mnnnn
d) Response : Memory display at terminal

2.1.3.2 Purpose

This function allows the user to dump successive memory locations, and modify them if desired.

2.1.3.3 Normal Processing

The command M0000-00FF dumps the memory locations from 0000 to FF with no chance for modification. The command M0010 <cr> displays the contents of location 10 (hex). If a second carriage return is typed, the contents of location 10 is unchanged and the contents of location of the next location is displayed. To change the location, type a new value (hex) before typing the second carriage return. A linefeed can be used to display and allow the modification of the previous memory location.

2.1.3.4 Abnormal Processing

After each location is written, it is read back and checked. If the 2 values do not match, the monitor leaves the modify memory function and prompts for a new command.

2.1.4 Display/Modify Registers

2.1.4.1 General Information

a) Function Category : Monitor
b) Function Name : REGISTERS
c) Activation : R or Rrxx command
d) Response : Register display at terminal

2.1.4.2 Purpose

This function allows the user to display or modify PCC processor registers.

2.1.4.3 Normal Processing

The R command causes all internal processor registers are displayed. The Rrxx command places the hex value xx into register r and then displays all registers.

2.1.5 Set/Clear Breakpoints

2.1.5.1 General Information

a) Function Category : Monitor
b) Function Name : BREAKPOINT
c) Activation : Bxxxx command
d) Response : A breakpoint is set at addr xxxx

2.1.5.2 Purpose

This function allows the user to set a breakpoint into his code.

2.1.5.3 Normal Processing

The user inserts a breakpoint into the code using the B command. A breakpoint can be cleared by setting it to an unused address. Each time a breakpoint is encountered while executing code, the user is returned to the debugger, the registers are dumped and the user has the option to continue (with the Go command) or to examine the machine environment before continuing.

When a breakpoint is encountered while executing code, all interrupts are disabled (real-time clock and I/O ports). Stepping thru code will enable I/O interrupts that were enabled when the breakpoint was encountered.

2.1.6 Arithmetic Function (HEX Calculator)

2.1.6.1 General Information

a) Function Category : Monitor
b) Function Name : ARITHMETIC
c) Activation : Axxxx+/-xxxx=
d) Response : Value of addition or subtraction

2.1.6.2 Purpose

This function allows the user to perform addition and subtraction of hex values.

2.1.6.3 Normal Processing

A0004+000F= 0013 (after = or <cr> the answer is printed)
A00FF-0001= 00FE

2.1.7 Echo/Noecho Terminal Input

2.1.7.1 General Information

a) Function Category : Monitor
b) Function Name : ECHO
c) Activation : E command
d) Response : Toggle echo state

2.1.7.2 Purpose

This function allows the user to enable or disable echoing of input from the terminal.

2.1.7.3 Normal Processing

If the current state of echoing is disabled, the echo command enables it, and vice-versa.

2.1.8 Execute Routine(GO)

2.1.8.1 General Information

a) Function Category : Monitor
b) Function Name : GO
c) Activation : Gxxxx command
d) Response : Jump to routine at addr xxxx

2.1.8.2 Purpose

This function allows the user to begin the execution of code which has been downloaded to the PCC.

2.1.8.3 Normal Processing

The address specified in the GO command is pushed on the stack, and an RTI instruction is executed, passing control to the routine. The default address used is the current value of the user program counter.

2.1.9 Display Current Time

2.1.9.1 General Information

a) Function Category : Monitor
b) Function Name : TIME
c) Activation : T command
d) Response : Current time displayed at terminal

2.1.9.2 Purpose

This function allows the user to display the current time contained in the PCC real-time clock.

2.1.9.3 Normal Processing

The time is displayed in the format: DD-MON-YY HR:MN:SC

2.1.10 Change Current Time

2.1.10.1 General Information

a) Function Category : Monitor
b) Function Name : CHANGE TIME
c) Activation : C command
d) Response : TT is prompted for new time/date

2.1.10.2 Purpose

This function allows the user to change the current time maintained in the PCC real-time clock.

2.1.10.3 Normal Processing

The terminal is prompted for new values for the time and date. The user can leave values unchanged by typing a return <cr> in response to the prompt for that value. Times must be entered using 24 hour (military) format and using hex (base 16) values.

2.1.11 Fill Memory

2.1.11.1 General Information

a) Function Category : Monitor
b) Function Name : FILL
c) Activation : Fmmmm-nnnn,xx

2.1.11.2 Normal Processing

All memory locations starting with mmmm and ending with nnnn are filled with the value xx.

2.1.12 Initialize

2.1.12.1 General Information

a) Function Category : Monitor
b) Function Name : INITIALIZE
c) Activation : I command or powerup routine
d) Response : on board LEDS indicate errors

2.1.12.2 Purpose

This function checks the validity of the PCC RAM and ROM and starts running the PCC operating system.

2.1.12.3 Normal Processing

The on board LED's indicate that power-up testing is underway. The LED's will light one at a time from DS1 to DS4 as each of the 4 tests is performed. RAM is checked by clearing and writing of all bits into the location to test for faulty bits. ROM is checked by performing a CRC operation on each ROM and comparing the calculated value to the value stored in each ROM.

2.1.12.4 Abnormal Processing

If any test indicates an error, the operator is notified by flashing two or more LEDs. The code displayed by the LED's is defined in Appendix G of this document.

2.1.13 Display Help Menu

2.1.13.1 General Information

a) Function Category : Monitor
b) Function Name : HELP
c) Activation : H command
d) Response : Help menu appears on terminal

2.1.13.2 Normal Processing

The help menu appears on the terminal. This menu illustrates all possible debugger commands and the appropriate syntax.

```
Axxxx+/-xxxx= : ARITHMETIC ie HEX CALULATOR (+ or -)
Bnnnn         : SET BREAKPOINT AT nnnn
C             : CHANGE TIME
D             : TOGGLE DEBUG SWITCH
E             : TOGGLE SERIAL TERMINAL ECHO FLAG
Fnnnn-mmmm,xx : FILL LOCATIONS nnnn to mmmm with xx
Gnnnn         : GO (EXECUTE CODE AT) LOCATION nnnn
H             : PRINT THIS HELP DISPLAY
I             : INITIALIZE PCC - SELF-CHECK AND START PCCOS
Jnnnn,mmmm    : SINGLE STEP SUBROUTINE FROM LOCATION nnnn
L             : LOAD (MIKBUG FORMAT)
Mnnnn         : OPEN MEMORY LOCATION nnnn FOR INSPECTION/MOD
Mnnnn-mmmm    : DUMP LOCATIONS nnnn TO mmmm
Rrnn          : MODIFY/DISPLAY REGISTER r
Snnnn,mmmm    : SINGLE STEP mmmm INSTRUCTIONS FROM LOCATION
T             : DISPLAY CURRENT TIME
Xnnnn-mmmm    : EXERCISE MEMORY LOCATIONS nnnn - mmmm
V             : VERIFY LOAD (MIKBUG FORMAT)
```

2.1.14 Single Step Debug

2.1.14.1 General Information

a) Function Category : Monitor
b) Function Name : STEP
c) Activation : S or J command
d) Response : Step to next intruction or execute next routine

2.1.14.2 Purpose

This mode of operation allows the programmer to step through code executing 1 or more instructions at a time and viewing their results.

2.1.14.3 Normal Processing

The S command executes the next instruction. The J command executes the entire next subroutine until control is returned into the main line code. The code to be executed is copied to a scratch area of RAM and followed by a SWI opcode. The code is executed by pointing the PC to this area and executing until the SWI returns control to the monitor. Either 1 instruction or a user selected number of instructions can be emulated in this manner.

2.1.14.4 Abnormal Processing

Certain operations are not emulated in the single step mode. They are treated as a NOP and cause a error message to be displayed.

```
PUL U,  **,PC
PSH S,  **,PC
PSH U,  **,PC
```

2.2 PCCOS

The PCC operating system contains general purpose routines to manage queues, dynamically allocate and deallocate message buffers, queue I/O requests to drivers, signal I/O completion. These routines are callable as Pascal functions so that the status of the functions can be checked. These routines are invoked via a SWI2 interrupt followed by a code to indicate the function to be performed and any arguments to be passed to the routine.

Semaphores are used within the PCC to indicate status and the occurrence of discrete events. 32 semaphores (1-32) are available for use by the personality module. Semaphores 33 - 64 are used by the operating system.

The operating system creates and maintains a group of queue elements which can be used by any component of the PCC. Queue elements available for use are linked into a list maintained by the operating system. To allocate a queue element for use a driver or program uses a PCCOS function. Deallocation is handled in a similar fashion, making the element available for use again. It is the responsibility of all software components to return queue elements to the available pool. The length of all queue elements is fixed, however their contents vary depending on use.

Queues are used by the PCC to synchronize communication with the K&S machine and the Factory Automation host. It is important to realize that the actual size of the queues will be small (no more than a few elements) unless there is break in the communication link. Queues are populated with queue elements which represent an I/O request to be transmitted or a message received which must be processed. In that situation, the queue of messages waiting to use the broken link will get as large as the PCC environment can support in an effort to preserve as much data as possible until the channel is reestablished. Depending on what types of messages are waiting (informational or requests for data) the channel may be repaired before either end is affected by the interruption. I/O queues contain elements which describe a message which must be output to the host or machine. Transaction queues contain elements which describe a command received from the machine or a SECS message received from the host which is to be processed.

PCCOS also creates and manages a group of message buffers which can be used by any component of the PCC to hold a SECS message. A driver would typically allocate one of the message buffers to hold an unsolicited input message. The driver would then allocate a queue element from the system-wide pool, insert in it the information needed to process the message just received, and then place this element in the appropriate queue. When the queue element had been processed, the SECS module would deallocate the message buffer and the queue element so that they can be reused.

2.2.1 Allocate Queue Element [ALLQEL]

2.2.1.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : ALLQEL
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.1.2 Purpose

This routine allocates a queue element from the pool of pre-defined elements.

2.2.1.3 Activation

Pascal:

```
FUNCTION ALLQEL(VAR ELEMENT:INTEGER):INTEGER;EXTERNAL;

STATUS := ALLQEL(ELEMENT);
IF STATUS <> ST_SUC THEN
  BEGIN

  END
ELSE ...
```

2.2.1.4 Function Input

a) ELEMENT is an integer variable which is to receive the address of the allocated queue element.

2.2.1.5 Function Output

a) ST_SUC - ELEMENT contains the address of the allocated element.
b) ST_ERR - The pool of available queue elements is empty. ELEMENT unchanged.

2.2.2 Deallocate Queue Element [DEAQEL]

2.2.2.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : DEAQEL
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.2.2 Purpose

This routine deallocates a queue element thus returning to the pool of available elements.

2.2.2.3 Activation

Pascal:

```
FUNCTION DEAQEL(VAR ELEMENT:INTEGER):INTEGER;EXTERNAL;

STATUS := DEAQEL(ELEMENT);
```

2.2.2.4 Function Input

a) ELEMENT is an integer variable containing the address of the queue element to be deallocated .

2.2.2.5 Function Output

a) ST_SUC - The element was successfully deallocated.
b) ST_ERR - ELEMENT was not in the range of valid queue elements. It was not returned to available pool.

2.2.3 Allocate Message Buffer [ALLMBF]

2.2.3.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : ALLMBF
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.3.2 Purpose

This routine allocates a message buffer from the pool of pre-defined buffers.

2.2.3.3 Activation

Pascal:

```
FUNCTION ALLMBF(VAR MSGBUF:INTEGER):INTEGER;EXTERNAL;

STATUS := ALLMBF(MSGBUF);
IF STATUS <> ST_SUC THEN
  BEGIN

  END
ELSE ...
```

2.2.3.4 Function Input

a) MSGBUF is an integer variable which is to receive the address of the allocated buffer.

2.2.3.5 Function Output

a) ST_SUC - MSGBUF contains the address of the allocated buffer.
b) ST_ERR - The pool of available message buffers is empty. MSGBUF unchanged.

2.2.4 Deallocate Message Buffer [DEAMBF]

2.2.4.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : DEAMBF
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.4.2 Purpose

This routine deallocates a message buffer thus returning it to the pool of buffers available for general use.

2.2.4.3 Activation

Pascal:

```
FUNCTION DEAMBF(VAR MSGBUF:INTEGER):INTEGER;EXTERNAL;

STATUS := DEAMBF(MSGBUF);
```

2.2.4.4 Function Input

a) MSGBUF is a variable which contains the address of the message buffer to be deallocated.

2.2.4.5 Function Output

a) ST_SUC - The message buffer was successfully deallocated.
b) ST_ERR - MSGBUF was not in the range of valid message buffers. It was not returned to available pool.

2.2.5 Insert Element Into Queue [INSQUE]

2.2.5.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : INSQUE
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.5.2 Purpose

This routine inserts an element into the queue specified. The insertion can occur at the head or the tail of the queue depending on parameters.

2.2.5.3 Activation

Pascal:

```
FUNCTION INSQUE(VAR QUEUE,ELEMENT:INTEGER;
                    LOCATION:INTEGER):INTEGER;EXTERNAL;

STATUS := INSQUE(QUEUE,ELEMENT,LOCATION);
```

2.2.5.4 Function Input

a) QUEUE is the list head address of the queue to be modified.
b) ELEMENT is the address of the element to be inserted.
c) LOCATION is a 0 to insert at tail of queue, or 1 to insert at head of queue.

2.2.5.5 Function Output

a) ST_SUC - The element was successfully inserted in queue.
b) ST_ERR - The element could not be inserted in the requested queue.

2.2.6 Remove Element From Head Of Queue [REMQUE]

2.2.6.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : REMQUE
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.6.2 Purpose

This routine removes an element from the head of the selected queue.

2.2.6.3 Activation

Pascal:

```
FUNCTION REMQUE(VAR QUEUE,ELEMENT:INTEGER):INTEGER;EXTERNAL

STATUS := REMQUE(QUEUE,ELEMENT);
IF STATUS <> ST_SUC THEN
  BEGIN

  END
ELSE ...
```

2.2.6.4 Function Input

a) QUEUE is the list head of the queue to be affected.
b) ELEMENT is a variable to receive the address of the element removed from the selected queue.

2.2.6.5 Function Output

a) ST_SUC - An element was removed from the specified queue. ELEMENT contains the address of the removed element.
b) ST_EMP - The specified queue was empty. ELEMENT is unchanged.

2.2.7 Set Semaphore [SETSEM]

2.2.7.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : SETSEM
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.7.2 Purpose

This routine sets a semaphore.

2.2.7.3 Activation

Pascal:

```
FUNCTION SETSEM(SEMAPHORE:INTEGER):INTEGER;EXTERNAL;

STATUS := SETSEM(SEMAPHORE)
```

2.2.7.4 Function Input

a) SEMAPHORE is an integer value of the semaphore to be set.

2.2.7.5 Function Output

a) ST_SUC - The specified semaphore was successfully set.
b) ST_ERR - The specified semaphore was invalid.

2.2.8 Reset Semaphore [RESSEM]

2.2.8.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : RESSEM
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.8.2 Purpose

This routine resets a semaphore.

2.2.8.3 Activation

Pascal:

```
FUNCTION RESSEM(SEMAPHORE:INTEGER):INTEGER;EXTERNAL;

STATUS := RESSEM(SEMAPHORE)
```

2.2.8.4 Function Input

a) SEMAPHORE is an integer value of the semaphore to be reset.

2.2.8.5 Function Output

a) ST_SUC - The specified semaphore was successfully reset.
b) ST_ERR - The specified semaphore was invalid.

2.2.9 Check Semaphore [CHKSEM]

2.2.9.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : CHKSEM
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.9.2 Purpose

This routine senses the condition of a semaphore.

2.2.9.3 Activation

Pascal:

```
FUNCTION CHKSEM(SEMAPHORE:INTEGER):INTEGER;EXTERNAL;

STATUS := CHKSEM(SEMAHPHORE);
IF STATUS = ST_SET THEN ...
ELSE ...;
```

2.2.9.4 Function Input

a) SEMAPHORE is an integer value of the semaphore to be reset.

2.2.9.5 Function Output

a) ST_SET - The specified semaphore was determined to be set.
b) ST_RES - The specified semaphore was determined to be reset.
c) ST_ERR - The specified semaphore was invalid.

2.2.10 Wait For Semaphore To Be Set [WAISEM]

2.2.10.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : WAISEM
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.10.2 Purpose

This routine senses the condition of the specified semaphore and returns control only when the semaphore is determined to be set. During testing, driver timeout processing and I/O interrupt processing continues.

2.2.10.3 Activation

Pascal:

```
FUNCTION WAISEM(SEMAPHORE:INTEGER):INTEGER;EXTERNAL;

STATUS := WAISEM(SEMAHPHORE);
IF STATUS = ST_SET THEN ...
ELSE ...;
```

2.2.10.4 Function Input

a) SEMAPHORE is an integer value of the semaphore to be reset.

2.2.10.5 Function Output

a) ST_SET - The specified semaphore was determined to be set.
b) ST_ERR - The specified semaphore was invalid.

2.2.11 Delay [DELAY]

2.2.11.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : DELAY
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.11.2 Purpose

This routine delays for the specified number of 100 millisecond intervals. During the wait time, interrupt processing continues. When the delay time expires, execution resumes at line of code following the delay.

2.2.11.3 Activation

Pascal:

```
FUNCTION DELAY(VALUE:INTEGER):INTEGER;EXTERNAL;

STATUS := DELAY(VALUE);
```

2.2.11.4 Function Input

a) VALUE is an integer value of the number of 100 milliseconds to wait for.

2.2.12 Set System Time [SETTIM]

2.2.12.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : SETTIM
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.12.2 Purpose

This routine sets the system time and date. Any parameters which are negative do not change the particular time quantity.

2.2.12.3 Activation

```
Pascal:
 TIME = RECORD
   YEAR:INTEGER;
   MONTH:INTGER;
   DAY:INTEGER;
   HOURS:INTEGER;
   MINUTES:INTEGER;
   SECONDS:INTEGER
   END;

   FUNCTION SETTIM(VAR TIME:INTEGER):INTEGER;EXTERNAL;

   STATUS := SETTIM(TIME);
```

2.2.12.4 Function Input

a) RECORD is the address of a user defined record containing the new time.
b) YEAR is the number of years since 1900 (1-99).
c) MONTH is the integer value of the month (1-12).
d) DAY is the day of the month (1-31).
e) HOURS are expressed in military time (0-23).
f) MINUTES are expressed in decimal (1-60).
g) SECONDS are expressed in decimal (1-60).

2.2.12.5 Function Output

a) ST_SUC - The system time was set.
b) ST_ERR - The time could not be set.

2.2.13 Get System Time [GETTIM]

2.2.13.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : GETTIM
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.13.2 Purpose

This routine returns the system time and date.

2.2.13.3 Activation

```
Pascal:
 TIME = RECORD
   YEAR:INTEGER;
   MONTH:INTGER;
   DAY:INTEGER;
   HOURS:INTEGER;
   MINUTES:INTEGER;
   SECONDS:INTEGER
   END;

   FUNCTION GETTIM(VAR TIME:INTEGER):INTEGER;EXTERNAL;

   STATUS := GETTIM(TIME)
```

2.2.13.4 Function Input

a) YEAR is the number of years since 1900 (1-99).
b) MONTH is the integer value of the month (1-12).
c) DAY is the day of the month (1-31).
d) HOURS are expressed in military time (0-23).
e) MINUTES are expressed in decimal (1-60).
f) SECONDS are expressed in decimal (1-60).

2.2.13.5 Function Output

a) ST_SUC - The system time was valid.
b) ST_ERR - The time was determined to be invalid.

2.2.14 Set Current Operating Mode [SETMOD]

2.2.14.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : SETMOD
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.14.2 Purpose

This routine sets the current operating mode of the PCC. The value is saved in battery-backed RAM so power failures do not affect it. The high order byte of the value is a read only location which defines the boolean status of communication with the host. A value of "true" indicates that communication is active and the PCC is online with the host , a value of "false" indicates communication problems with the host. Whenever the high byte of the mode value is false, the PCC attempts to re-establish communication with the host periodically. The low order byte of the mode value is available to the personality module to indicate any mode-related quantity.

2.2.14.3 Activation

Pascal:

```
FUNCTION SETMOD(VAR VALUE:INTEGER):INTEGER;EXTERNAL;

STATUS := SETMOD(VALUE);
```

2.2.14.4 Function Input

a) MODE is value of the new operating mode. Only the low order byte of the mode word can be changed.

2.2.14.5 Function Output

a) ST_SUC - Success always returned.

2.2.15 Get Current Operating Mode [GETMOD]

2.2.15.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : GETMOD
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.15.2 Purpose

This routine returns the current operating mode of the PCC. The value is saved in battery-backed RAM so power failures do not affect it. The high order byte of the value is a read only location which defines the boolean status of communication with the host. A value of 'true' indicates that communication is active and the PCC is online with the host , a value of 'false' indicates communication problems with the host. Whenever the high byte of the mode value is false, the PCC attempts to re-establish communication with the host.

2.2.15.3 Activation

Pascal:

```
FUNCTION GETMOD(VAR VALUE:INTEGER):INTEGER;EXTERNAL;

STATUS := GETMOD(VALUE);
```

2.2.15.4 Function Input

a) MODE is value of the new operating mode.

2.2.16 Restart The PCC [RESTRT]

2.2.16.1 General Information

a) Function Category : PCCOS Runtime Support
b) Function Name : RESTRT
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.2.16.2 Purpose

This function performs a warm restart of the PCC. It performs all normal power-up functions such as initialzing queues, allocating elements and buffers, and setting up drivers. It does not perform the ROM and RAM tests which are only performed during power-up.

2.2.16.3 Activation

Pascal:

```
FUNCTION RESTRT():INTEGER;EXTERNAL;

STATUS := RESTRT();
```

2.3 I/O DRIVERS

At startup, all PCC drivers are set up to ignore unsolicited input. The initialization program in the personality module (called as part of startup) enables the appropriate driver(s) to accept unsolicited input.

Most incomming messages to the PCC (from the host and the bonder) are handled by drivers as unsolicited input, ie. the message is received and placed in a message buffer which was allocated from the list of pre-defined message buffers maintained by the monitor. A queue element is then allocated by the driver from the list of pre-defined queue elements, filled in with the information necessary to process the message just received, and placed into the transaction queue of messages initiated from the host where it will be processed. All outgoing messages from the PCC are initiated by the SECS module, and will be sent in the order in which they were queued for transmission.

For communication to the host, the IO function normally inserts an I/O element at the tail of a queue which contains all elements waiting to use the particular channel. If the queue is empty, the driver is entered at it's call entry point. This code sets up the device to handle the just queued I/O element and returns to the monitor dispatcher. The actual I/O is performed by the interrupt section of the driver. There are also situations where a machine requires an immediate answer to a request for information. In this case the IO function is initiated using a function code to indicate a high priority I/O request. The I/O element is placed at the head of the queue so that it is next to be sent. Write requests return control immediately to the caller. Read requests return control when the requested data is available.

Routines which initiate I/O requests have two means available to determine when the I/O actually completes. The routine can specify a completion semaphore to be set at I/O completion. If this method is used, the semaphore is automatically reset when the I/O is initiated and becomes set when the I/O request completes (either successfully, unsucessfully or times out). The second method is to specify a completion routine to execute when the I/O completes (either successful, unsuccessful or timeout). The completion routine can determine the completion status of the I/O by examining the high order byte of the function location in the I/O element.

SECS protocol specifies that all messages and protocol exchanges be timed by software settable timers. T1 is the time between individual characters of an exchange, T2 is the time between protocol handshakes and T3 is the maximum time to wait for a reply to a request for data. These timers are all associated with I/O on the SECS channel and so are implemented as part of the I/O element structure. Regardless of the physical means of communication, the calling program utilizes the same IO interface.

2.3.1 QUEUE AN I/O ELEMENT TO A DEVICE

2.3.1.1 General Information

a) Function Category : Pascal I/O Interface
b) Function Name : IO
c) Activation : Pascal Function
d) Response : integer status returned to caller

2.3.1.2 Purpose

This routine allocates a queue element and places information into it indicating how and to which device the I/O is take place.

2.3.1.3 Activation

Pascal:

```
FUNCTION IO(DEV,FUNC:INTEGER; VAR BUF:BUFPNT; CNT,SEM:INTEGER;
          VAR CMPADR:INTEGER; TIMOUT:INTEGER):INTEGER;EXTERNAL
```

```
STATUS :=IO(DEV,FUNC,BUF,CNT,SEM,CMPADR,TIMOUT);
IF STATUS <> ST_SUC THEN
  BEGIN

  END
```

2.3.1.4 Function Input

a) DEV is a byte containing the device code of the device to/from the I/O is to be performed (eg. IO_SR,IO_SX).
b) FUNC is an integer describing the desired I/O function (eg. IO_RD,IO_WRT...). The low byte describes the operation (ie. read,write). The high byte modifies the function (ie. high priority, disable timeout).
c) TIMOUT is the time in 100 millisecond increments which can elapse from the time the I/O starts until it completes (maximum 25 seconds).
d) BUF is the address of the buffer.
e) CNT is the byte count of the data to be transferred.
f) SEM is a semaphore to be set at completion of the I/O. This semaphore is reset by the I/O processor when the I/O starts.
g) CMPADR is the address of a routine to be executed at I/O completion. This routine is executed at interrupt level (however not blocking single character I/O on the SECS port). It should be coded to execute quickly, and do a minimum of processing.

Message buffers queued to PCC I/O drivers must follow the following buffer layouts. The SECS module provides a function to generate a SECS formatted message from equipment format, and vice versa. (SEE SECFMT - EQPFMT)

SECS IO

R	EQUIPMENT ID	
W	STREAM	FUNCTION
E	BLOCK NUMBER	
	SYSTEM BYTES	
	SYSTEM BYTES	
	DATA	

EQUIPMENT FORMAT
SHARED RAM/PARALLEL PORT I/O

CONTROL	DATLEN*
STREAM	FUNCTION
BLOCK NUMBER	
DATA .... (datlen bytes)	

* DATLEN is the length of data only. It does not include the control information (stream, function, block number etc).

2.3.1.5 Function Output

a) ST_SUC - the I/O element was sucessfully queued.
b) ST_ERR - the completion semaphore was invalid.
c) ST_IDV - an invalid device was specified.
d) ST_IFN - an invalid function was specified.
e) ST_IBF - an invalid message buffer was specified.
f) ST_EMP - no queue elements are available.
g) ST_RTM - the equipment timed out the I/O request.
h) ST_ABO - the equipment aborted the I/O request.

2.3.1.6 Completion Status - (returned In High Order Byte Of Function)

a) IS_SUC - the I/O completed sucessfully.
b) IS_ERR - the I/O completed with errors.
c) IS_TIM - the I/O timed out.
d) IS_RTY - the I/O exhausted the maximum retry count, but was not successful.

2.3.2 SECS Driver (SX)

The SECS port is used to communicate with the host computer system. The following I/O functions are supported:

1. IO_WRT Queue an I/O element to be sent to the host. The SECS message is queued to be sent to the host and control returns immediately to the caller. The I/O element is placed at the end of the queue.

    1. Modifier IM_PRI places the element at top of queue.

    2. Modifier IM_RPY starts a timer to insure that the reply is received within the SECS T3 time.

    3. Modifier IM_NOTIM disables timeout processing for this I/O request.

    4. Modifier IM_NODEA disables deallocation of resources when request completes.

2. IO_RD Queue an outstanding read for the next SECS message received.

3. IO_SET Set a device constant. The constant to be set is placed in the CNT variable of the I/O request and the new value of the constant is placed in the TIMOUT variable. These values are stored in non-volatile RAM so they need not be set at each powerup or PCC restart.

    1. DC_T1 - SECS T1 (intercharacter timer units are for DUART counter timer).

    2. DC_T2 - SECS T2 (protocol timer units = 100 millisec).

    3. DC_T3 - SECS T3 (reply timer units = 100 millisec).

    4. DC_RTY- SECS retry count.

    5. DC_SBR- SECS baud rate (value is in DUART codes - see appendix E for values).

    6. DC_IBT- Interblock delay between successive blocks of a multi-block message (units = 100 millisec).

4. IO_ENA Enable unsolicited input. Any messages received while no read is active, are placed in the host transaction queue.

5. IO_DIS Disable unsolicited input.

6. IO_ABO Abort all instances of the specified stream/function from SECS queue. CNT contains the stream to be aborted.

2.3.3 Shared RAM Driver (SR)

Note that in the discussion below, the IO_WRT and IO_RD functions are the only ones that are queued. All other operations are completed before returning to the caller.

1. IO_WRT Queue an I/O element to be sent to the machine. The data is queued to be sent to the machine and control returns immediately to the caller. The I/O element is placed at the end of the queue.

    1. Modifier IM_PRI places the element at head of queue.

    2. Modifier IM_NOTIM disables the use of timeouts for this I/O request.

    3. Modifier IM_NODEA disables deallocation of resources when request completes.

2. IO_RD Read the contents of the machine's write area in shared RAM. The I/O completes when the machine indicates that a message is ready in it's write area or the request times out.

3. IO_TIM Indicate to the equipment that an expected reply from the FAS host was not received within the SECS T3 time. The expected stream is placed in the IO LEN argument, the expected function is placed in the TIMOUT argument.

4. IO_ENA Enable unsolicited input. Any messages received while no read is active, are placed in the host transaction queue.

5. IO_DIS Disable unsolicited input.

2.3.4 Parallel Port Driver (PP)

Implementation note: Because the bonder controls all communication with the PCC in a parallel port configuration, the concept of I/O timeout is not supported for the parallel port driver.

1. IO_ENA Enable unsolicited input. Any messages received while no read is active, are placed in the host transaction queue.

2. IO_DIS Disable unsolicited input.

3. IO_RD Queue a read request to the bonder. The request will complete when the next message is received from the bonder.

4. IO_WRT Queue a write request to the bonder. The I/O completes when the bonder issues a 'polled read' to the PCC.

1. Modifier IM_PRI places the element at head of queue.

2. Modifier IM_NODEA disables deallocation of resources when request completes.

2.3.5 Barcode Driver (BC Or TT)

1. IO_RD Read a block of data.

2. IO_WRT Write a block of data.

3. IO_ENA Enable unsolicited input.

NOT YET IMPLEMENTED

2.4 SECS MODULE

The SECS module handles all control and synchronization functions within the PCC. All input and output is controlled by the SECS module based on SECS II protocol and it's knowledge of K&S machines.

2.4.1 Node Transaction Protocol(SECS II)

The transaction protocol, implemented in the SECS module, has three main functions. First, it coordinates the receipt of multiblock messages. Second, it provides the mechanism for matching prime and secondary messages (ie. requests for data, and their responses). Third, it performs error detection on incomplete message transactions whether due to a block received out of order or a timeout occuring on a prime message which expected a reply (SECS T3 timer).

Message processing and generation utilities are available within the SECS module which simplify the processing and generation of messages. These routines reside in the SECS module, and are invoked by code in the personality module by use of a SWI3 interrupt and jump table in the SECS module. This technique allows the routines in the personality module to be linked independantly of the utility routines. These utility routines, like the PCCOS support routines, are available for use in all PCC implementations.

2.4.2 Cancel Current Message [CANMSG]

2.4.2.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : CANMSG
c) Activation : Pascal function

2.4.2.2 Purpose

This routine informs the SECS module to cancel further processing of the specified message. In response to this directive, the SECS module removes entries which are continuations of the specified message from the current queue it is servicing and throws them away. Any outstanding response timers for this stream are canceled.

2.4.2.3 Activation

Pascal:

```
PROCEDURE CANMSG(STREAM:INTEGER);EXTERNAL;

CANMSG(STREAM);
```

2.4.2.4 Function Input

a) STREAM is the value of SECS stream to be canceled.

2.4.3 Format SECS Message [SECFMT]

2.4.3.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : SECFMT
c) Activation : Pascal Procedure

2.4.3.2 Purpose

This routine converts the equipment message in the buffer to SECS format. If the message is a response to a host inquiry, the saved system bytes are inserted into the header along with the equipment ID. The length returned is determined from the DATLEN location in the equipment format message currently in the buffer.

2.4.3.3 Activation

Pascal:

```
PROCEDURE SECFMT(VAR MSGBUF:BUFPNT;
                 VAR BUFLEN:INTEGER);EXTERNAL;

SECFMT(BUFADR,LENGTH);
```

2.4.3.4 Function Input

a) MSGBUF (VAR) is the address of the buffer containing the message data.
b) LENGTH (VAR) is the address of a variable to receive the length of the message returned.

2.4.3.5 Function Output

a) MSGBUF contains the address of the message buffer. The data contained in the buffer is now represented in the SECS FORMAT detailed in the discussion the IO function.
b) LENGTH is returned containing the length of the SECS message. This length includes the header, all data bytes and the checksum (but not the length byte). This length may be passed to the SECS driver as the I/O length.

2.4.4 Format Equipment Message [EQPFMT]

2.4.4.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : EQPFMT
c) Activation : Pascal Procedure

2.4.4.2 Purpose

This routine converts the SECS message in the buffer to equipment format. If the message is a response to a host inquiry, the saved system bytes are inserted into the header along with the equipment ID.

2.4.4.3 Activation

Pascal:

```
PROCEDURE EQPFMT(VAR MSGBUF:BUFPNT;
                 VAR BUFLEN:INTEGER);EXTERNAL;

EQPFMT(MSGBUF,LENGTH);
```

2.4.4.4 Function Input

a) MSGBUF (VAR) is the address of the buffer containing the message data.
b) LENGTH is the length of the SECS message currently in the buffer. This length is the value returned by the SX driver and includes the header, data and the checksum.

2.4.4.5 Function Output

a) MSGBUF contains the address of the message buffer. The data contained in the buffer is now represented in the EQUIPMENT FORMAT detailed below:
b) BUFLEN contains the entire length of the equipment formatted message. It includes the datlen, strm, func ,block number and all data bytes. This length may be passed to the SR or PP drivers as the I/O length.

2.4.5 Build SECS Message [SECBLD]

2.4.5.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : SECBLD
c) Activation : Pascal Procedure

2.4.5.2 Purpose

This routine completes the building of a SECS format message from the incomplete one currently contained in the message buffer. If the message is a response to a host inquiry, the saved system bytes are inserted into the header along with the equipment ID.

2.4.5.3 Activation

Pascal:

```
PROCEDURE SECBLD(VAR MSGBUF:BUFPNT;
                 VAR BUFLEN:INTEGER);EXTERNAL;

SECBLD(BUFADR,LENGTH);
```

2.4.5.4 Function Input

a) MSGBUF (VAR) is the address of the buffer containing the message data.
b) LENGTH (VAR) is the address of a variable which contains the current length of the incomplete SECS message. It includes the entire header and all data.

2.4.5.5 Function Output

a) MSGBUF contains the address of the message buffer. The data contained in the buffer is now represented in the SECS FORMAT detailed in the discussion the IO function.
b) LENGTH is returned containing the length of the entire SECS message. This length includes the header, all data bytes and the checksum (but not the length byte). This length may be passed to the SECS driver as the I/O length.

2.4.6 Put Item Header Into Buffer [PUTIHD]

2.4.6.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : PUTIHD
c) Activation : Pascal Procedure

2.4.6.2 Purpose

This routine inserts an item header into the output message buffer.

2.4.6.3 Activation

Pascal:

```
PROCEDURE PUTIHD(ITMFMT,ITMSIZ:INTEGER;
 VAR OFFSET:INTEGER;VAR MSGBUF:BUFPNT);EXTERNAL;

 PUTIHD(ITMFMT,ITMSIZ,OFFSET,MSGBUF);
```

2.4.6.4 Function Input

a) ITMFMT is the format of the item.
b) ITMSIZ is the size of the item.
c) OFFSET (VAR) is a pointer to the last filled byte of the data area in the output buffer.
d) MSGBUF (VAR) is the address of the output message buffer.

2.4.6.5 Function Output

a) MSGBUF has had the item inserted.
b) POINTER has been updated to the last used byte in the message buffer.

2.4.7 Put Integer Data Into Buffer [PUTINT]

2.4.7.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : PUTINT
c) Activation : Pascal Procedure

2.4.7.2 Purpose

This routine inserts a 2 BYTE integer into the output message buffer.

2.4.7.3 Activation

```
PROCEDURE PUTINT(FMTSIZ,VALUE:INTEGER;
          VAR MSGBUF:BUFPNT);EXTERNAL;

PUTINT(FMTSIZ,VALUE,OFFSET,MSGBUF);
```

2.4.7.4 Function Input

a) FMTSIZ is the size of the integer to insert (1 or 2 bytes).
b) VALUE is the data value to be inserted.
c) OFFSET (VAR) is a pointer to the last filled byte of output buffer.
d) MSGBUF (VAR) is the address of the output message buffer.

2.4.7.5 Function Output

a) MSGBUF has had the item inserted.
b) OFFSET has been updated to the last used byte in the message buffer.

2.4.8 Put 4 Byte Integer Into Buffer [PUTIN4]

2.4.8.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : PUTIN4
c) Activation : Pascal Procedure

2.4.8.2 Purpose

This routine inserts a 4 BYTE integer into the output message buffer.

2.4.8.3 Activation

```
PROCEDURE PUTIN4(FMTSIZ,VALHI,VALLO:INTEGER;
          VAR MSGBUF:BUFPNT);EXTERNAL;

PUTIN4(FMTSIZ,VALUE,OFFSET,MSGBUF);
```

2.4.8.4 Function Input

a) FMTSIZ is the size of the integer to insert (4 bytes).
b) VALHI is the most significant data value to be inserted.
c) VALLO is the least significant data value to be inserted.
d) OFFSET (VAR) is a pointer to the last filled byte of output buffer.
e) MSGBUF (VAR) is the address of the output message buffer.

2.4.8.5 Function Output

a) MSGBUF has had the item inserted.
b) OFFSET has been updated to the last used byte in the message buffer.

2.4.9 Put Characters Into Buffer [PUTCHR]

2.4.9.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : PUTCHR
c) Activation : Pascal Procedure

2.4.9.2 Purpose

This routine inserts a string of bytes into the output message buffer.

2.4.9.3 Activation

```
PROCEDURE PUTCHR(ITMSIZ:INTEGER;
 VAR DATBYT:ARRAY[1..244]OF CHAR;
 VAR OFFSET:INTEGER;VAR MSGBUF:BUFPNT);EXTERNAL;

 PUTCHR(ITMSIZ,DATBYT,OFFSET,MSGBUF);
```

2.4.9.4 Function Input

a) ITMSIZ is the size of the item.
b) DATBYT is the address of an array containing the characters.
c) OFFSET (VAR) is a pointer to the last filled byte of output buffer.
d) MSGBUF (VAR) is the address of the output message buffer.

2.4.9.5 Function Output

a) MSGBUF has had the item inserted.
b) OFFSET has been updated to point to the last used byte in the message buffer.

2.4.10 Put Binary Data Into Buffer [PUTBIN]

2.4.10.1 General Information

a) Function Category : Message Processing Utilities
b) Function Name : PUTBIN
c) Activation : Pascal Procedure

2.4.10.2 Purpose

This routine inserts binary data into the output message buffer.

2.4.10.3 Activation

```
PROCEDURE PUTBIN(ITMSIZ:INTEGER;
 VAR DATBYT:ARRAY[1..244]OF CHAR;
 VAR OFFSET:INTEGER;VAR MSGBUF:BUFPNT);EXTERNAL;

 PUTBIN(ITMSIZ,DATBYT,OFFSET,MSGBUF);
```

2.4.10.4 Function Input

a) ITMSIZ is the size of the item.
b) DATBYT is the address of an array containing the binary data.
c) OFFSET (VAR) is a pointer to the last filled byte of output buffer.
d) MSGBUF (VAR) is the address of the output message buffer.

2.4.10.5 Function Output

a) MSGBUF has had the item inserted.
b) OFFSET has been updated to point to the last used byte in the message buffer.

2.5 PERSONALITY MODULE

The Personality Module consists of data tables and routines which are unique to a particular K&S machine.

2.5.1 Personality Module Configuration And Entry Table

The first 128 bytes of the personality module are reserved for entry points to the routines contained within that module. The first entry is for the initialization routine to be executed at startup. This routine enables the appropriate drivers and sets up necessary counters and variables in PCC RAM. The following table entries point to command processing routines, message processing routines and message generation routines. The purpose of these routines is discussed in the following sections. Appendix F lists the additional entry points contained in this table.

The personality module entry table also contains default parameters to allow customization of the PCC. Baud rate, timeout values, layout of shared RAM and retry counts are defined in this table.

2.5.2 Command Processing Routine

The personality module contains a Pascal routine to process all commands which can be received from the machine. The routine contains a section of code for each individual command to be received and knows what actions are to be performed for this command. Typical actions would be to update a counter maintained in PCC RAM or to send a SECS message to the Factory Automation host. Note that even though the arguments for the command processing routine are declared as integers, the values are 8 bit values and reside in the low 8 bits of the quantity. The command processing routine should be defined as shown below:

```
PROCEDURE COMMAND(CMD,CDATA1,CDATA2,CDATA2,CDATA3,CDATA4:INTEGER);
```

2.5.3 Host Message Handling Routines

The personality module contains a Pascal routine to process all functions for a particular SECS stream which may be received from the host. Each message processor contains a section of code for each SECS function to be performed and contains enough knowledge of the particular machine to gather any necessary data. In particular, the stream interpreting routine knows the appropriate type of communication to initiate with the bonder (if any at all) or where within the PCC the data resides. Note that the length is the entire length of the message including header, data and checksum.
Host message handling routines must be defined as shown below:

```
PROCEDURE HOST_STRM1(VAR BUFADR:BUFPNT;VAR LENGTH:INTEGER);
```

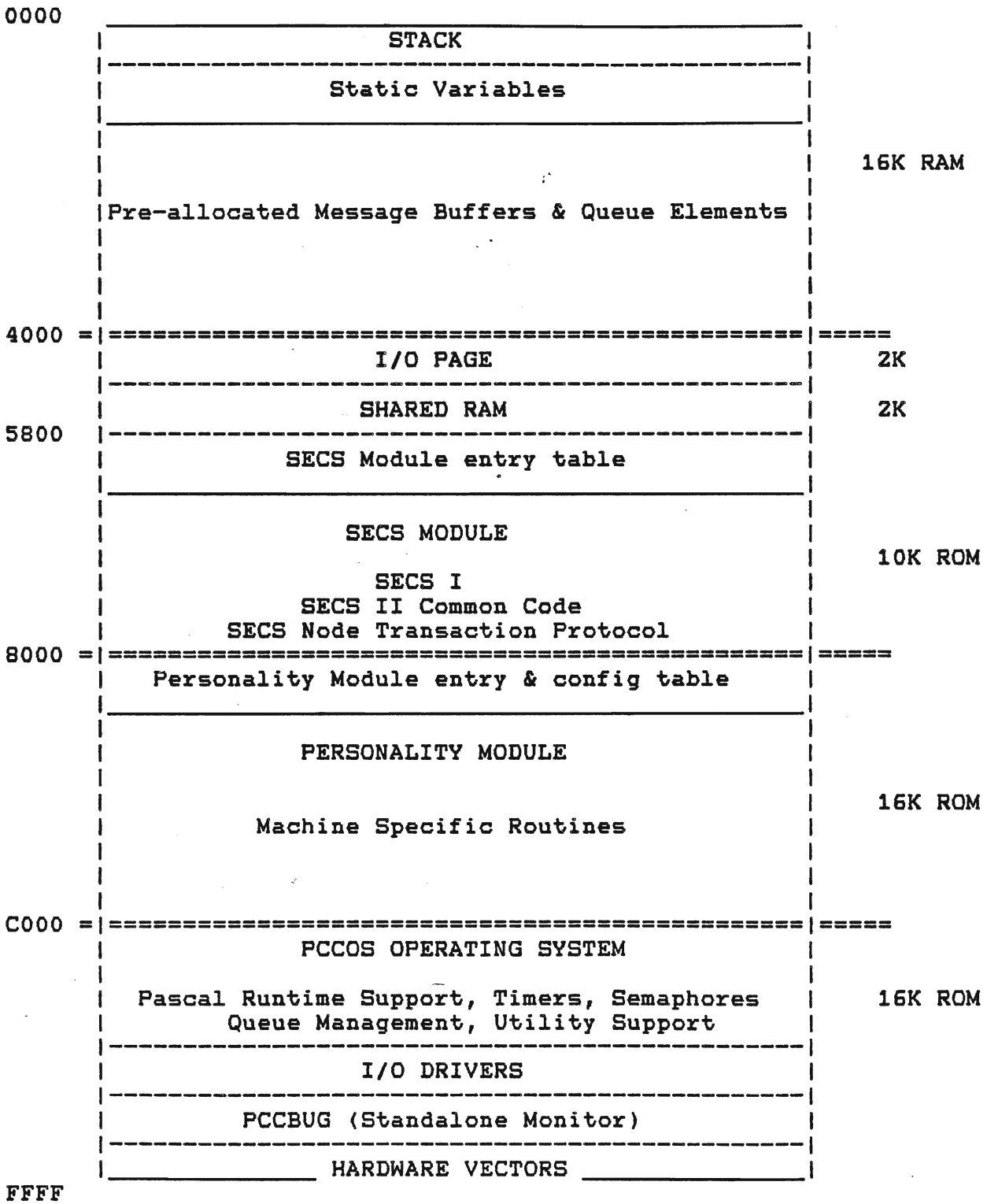
2.5.4 Equipment Message Handling Routines

The personality module contains a Pascal routine to handle the appropriate SECS message for each function of a particular SECS stream which the equipment can send to the host. Note that the length is the entire length of the message including header and data.
Equipment message handling routines must be defined as shown below:

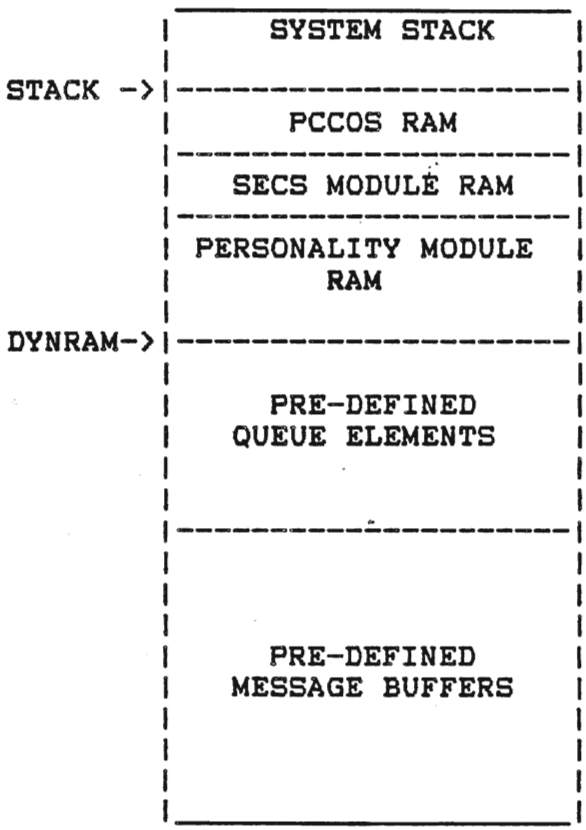
```
PROCEDURE EQUIP_STRM1(VAR BUFADR:BUFPNT;VAR LENGTH:INTEGER);
```

APPENDIX A

PCC MEMORY MAP

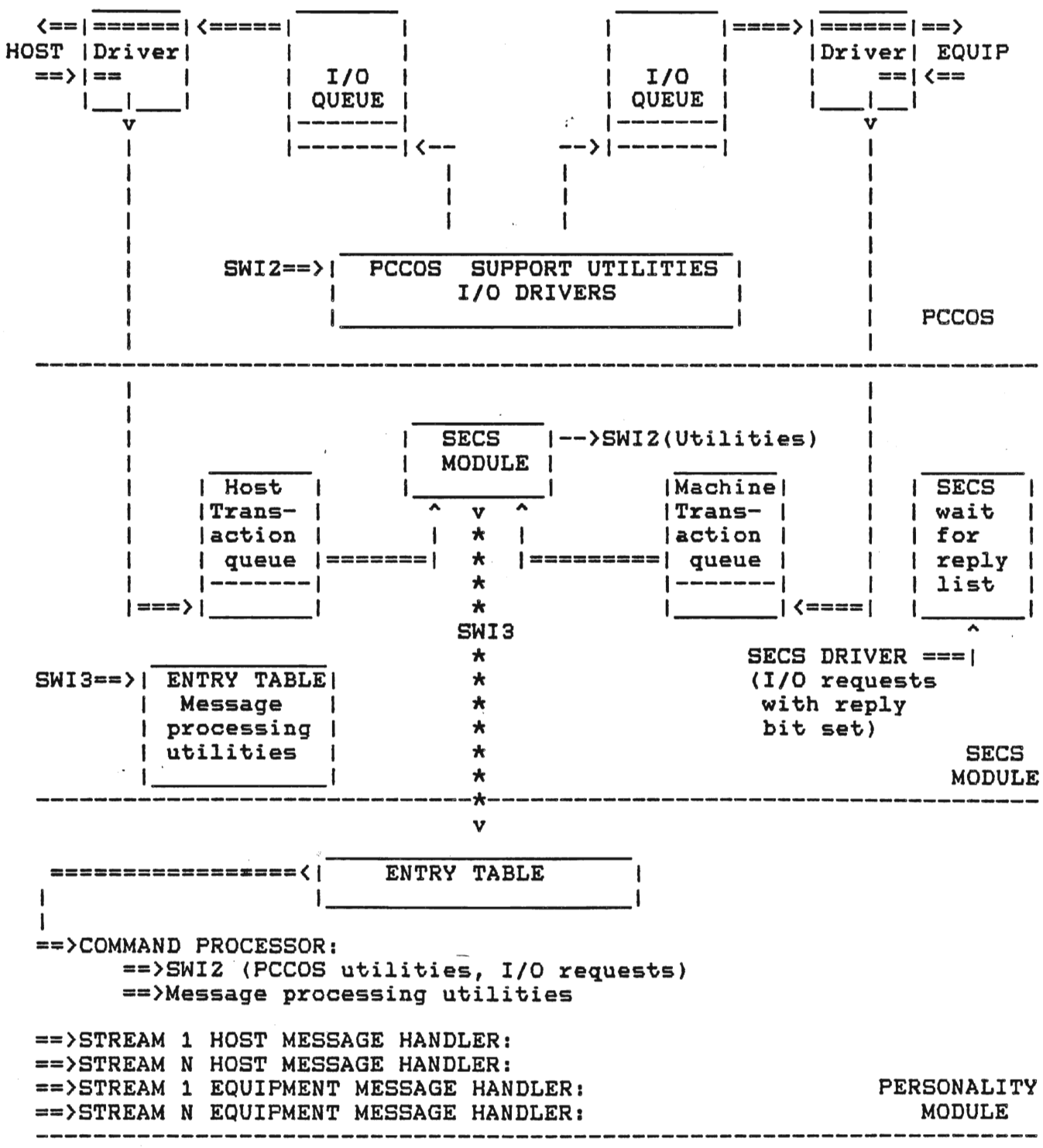

0000
STACK
Static Variables
16K RAM
Pre-allocated Message Buffers & Queue Elements
4000
I/O PAGE
2K
SHARED RAM
2K
5800
SECS Module entry table
SECS MODULE
10K ROM
SECS I
SECS II Common Code
SECS Node Transaction Protocol
8000
Personality Module entry & config table
PERSONALITY MODULE
16K ROM
Machine Specific Routines
C000
PCCOS OPERATING SYSTEM
Pascal Runtime Support, Timers, Semaphores
Queue Management, Utility Support
16K ROM
I/O DRIVERS
PCCBUG (Standalone Monitor)
HARDWARE VECTORS
FFFF

PCC RAM LAYOUT



APPENDIX B

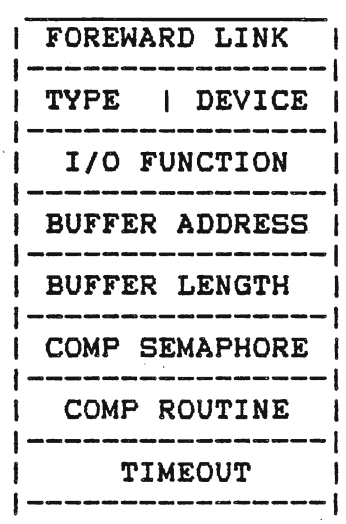
BLOCK DIAGRAM OF PCC SOFTWARE


HOST
Driver
I/O QUEUE
I/O QUEUE
Driver
EQUIP
SWI2==>
PCCOS SUPPORT UTILITIES
I/O DRIVERS
PCCOS
SECS MODULE
-->SWI2(Utilities)
Host Transaction queue
Machine Transaction queue
SECS wait for reply list
SWI3
SWI3==>
ENTRY TABLE
Message processing utilities
SECS DRIVER ===|
(I/O requests with reply bit set)
SECS MODULE
ENTRY TABLE
==>COMMAND PROCESSOR:
==>SWI2 (PCCOS utilities, I/O requests)
==>Message processing utilities
==>STREAM 1 HOST MESSAGE HANDLER:
==>STREAM N HOST MESSAGE HANDLER:
==>STREAM 1 EQUIPMENT MESSAGE HANDLER:
==>STREAM N EQUIPMENT MESSAGE HANDLER:
PERSONALITY MODULE

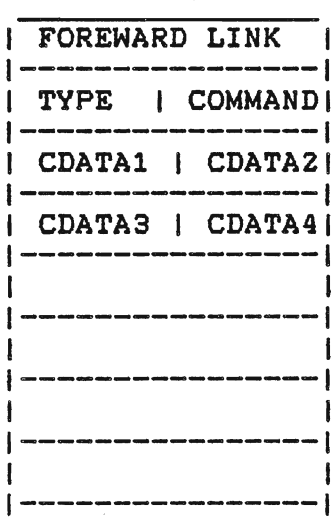
APPENDIX C

QUEUE ELEMENT LAYOUT

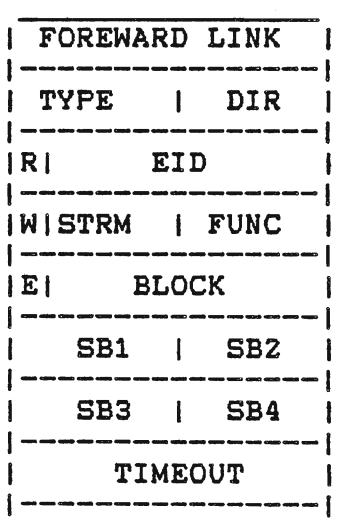
I/O ELEMENT



COMMAND TRANSACTION ELEMENT

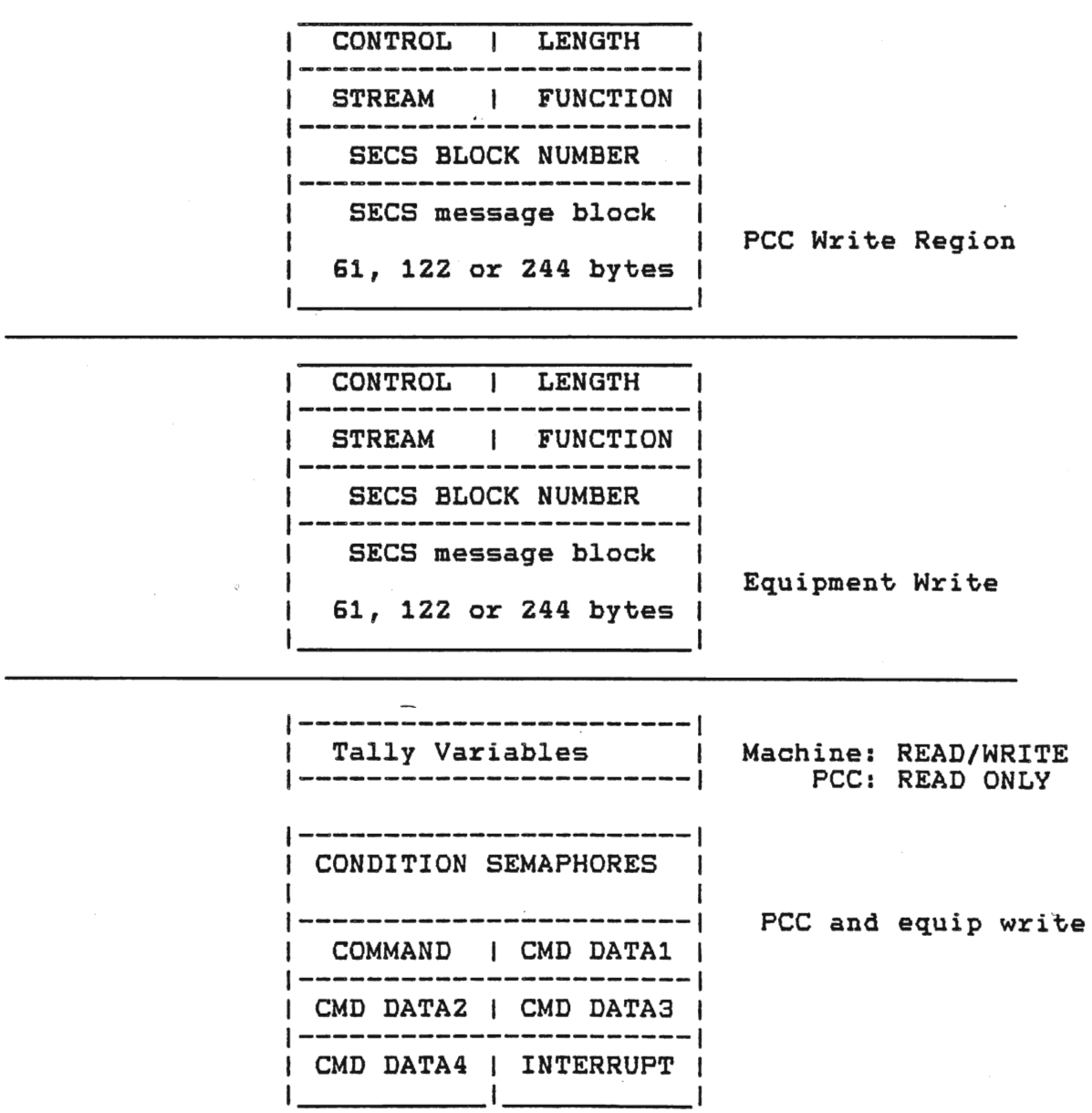


SECS NODE TRANSACTION ELEMENT



APPENDIX D

MEMORY MAP OF SHARED RAM



CONDITION SEMAPHORES

EQUIPMENT			PCC	
RCV	RW	DATA READY <--	WO	TRAN
RCV	WO	DATA RCVD -->	RW	TRAN
RCV	RW	TIMEOUT <--	WO	TRAN
RCV	WO	ABORT -->	RW	TRAN
RCV	RW	SPARE1 <--	WO	TRAN
RCV	WO	SPARE2 -->	RW	TRAN
TRAN	WO	DATA READY -->	RW	RCV
TRAN	RW	DATA RCVD <--	WO	RCV
TRAN	WO	TIMEOUT -->	(not implemented)	
TRAN	RW	ABORT <--	WO	RCV
TRAN	WO	SPARE1 -->	RW	RCV
TRAN	RW	SPARE2 <--	WO	RCV

TRUE=FF ; FALSE = 00 ; WO = WRITE ONLY ; RW = READ/WRITE
AT POWERUP ALL SEMAPHORES ARE RESET TO FALSE BY THE PROCESSOR
WHICH HAS RW ACCESS.

CONTROL BYTE LAYOUT

The control byte indicates the status of the owner's write area in shared RAM.

APPENDIX E

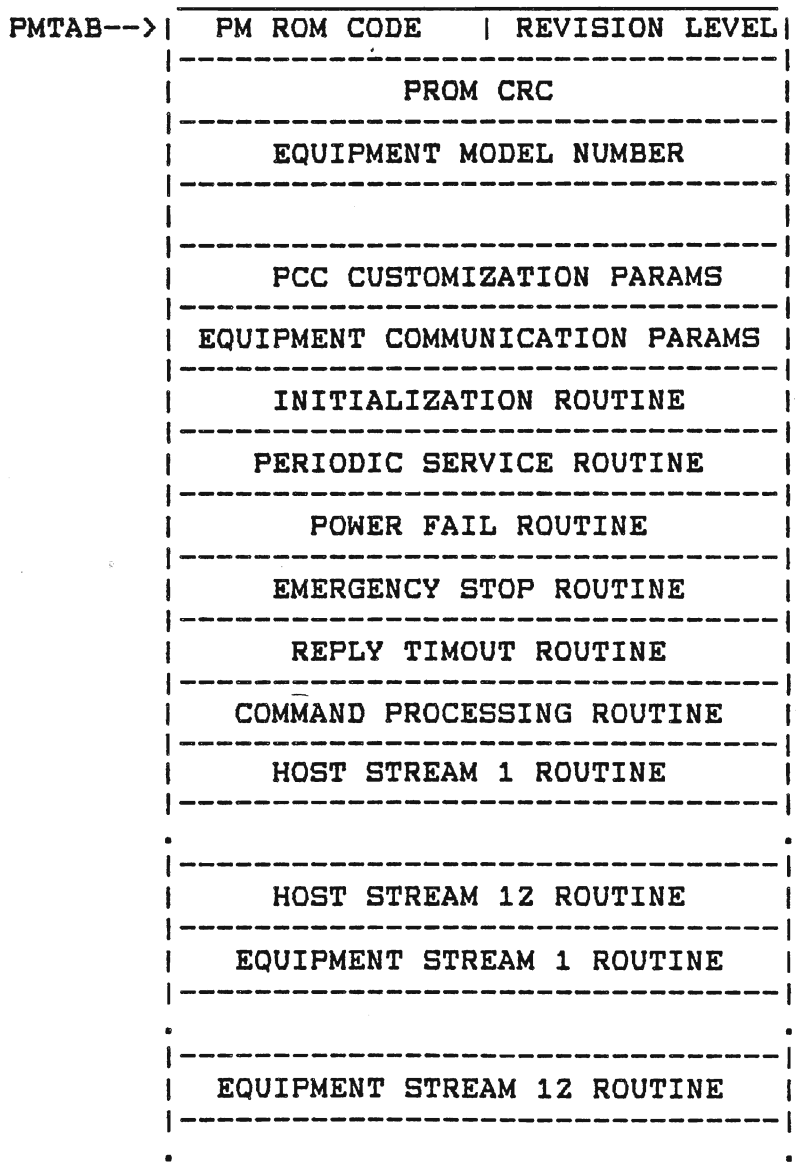
GLOBAL SYMBOL DEFINITION CONVENTION

I/O Devices	DV_XXX		DV_BC	4	Barcode Reader
			DV_SR	2	Shared RAM
			DV_SX	1	SECS Port
			DV_PP	3	Parallel Port
I/O Functions (low order byte)	IO_XXX	word	IO_RD	2	Read
			IO_WRT	1	Write
			IO_ENA	4	Enable unsolicited input
			IO_DIS	5	Disable unsolicited input
			IO_ABO	6	Abort instances of stream
I/O Modifiers (high order byte)	IM_XXX	bitmsk	IM_PRI	40	High Priority
			IM_RPY	80	Reply required
			IM_NOT	20	Disable timeouts for reques
			IM_NOD	10	No deallocation of resource
IO Status	IS_XXX	byte	IS_ABO	5	I/O request aborted
			IS_ERR	0	Error completion
			IS_RTM	4	remote time out by equip
			IS_SUC	255	Successful completion
			IS_TIM	3	I/O timed out
PCCOS system service status	ST_XXX	word	ST_SUC	255	Successful completion
			ST_ERR	0	Error completion
			ST_EMP	5	Queue empty
			ST_RES	1	Semaphore reset
			ST_SET	2	Semaphore set
			ST_IDV	3	Illegal device
			ST_IFN	4	Illegal function
			ST_IDE	6	Illegal element
			ST_IBF	7	Illegal buffer
Driver const	DC_XXX	byte	DC_T1	1	SECS T1 timer
			DC_T2	2	SECS T2 timer
			DC_T3	3	SECS T3 timer
			DC_IBT	4	Interblock timer
			DC_SBR	6	SECS baud rate
			DC_RTY	5	Retry count

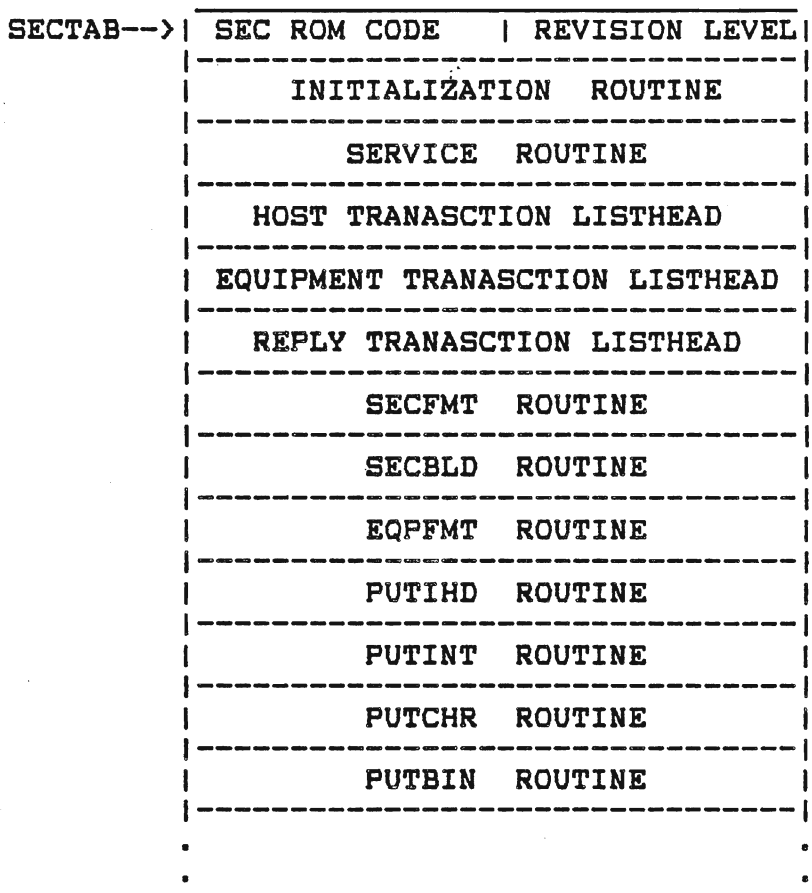
APPENDIX F

ENTRY TABLES

PERSONALITY MODULE ENTRY TABLE



SECS MODULE ENTRY TABLE



APPENDIX G

HARDWARE DETAILS

LED DEFINATION DURING NORMAL AND ERROR CONDITIONS

During normal operation, LEDS indicate the following conditions:

LED DS1 - Blinks at .5 second frequency as real time clock ticks.
LED DS2 - Communication status
(ON indicates problem - possibly host down)
(OFF indicates communication OK)
LED DS3 - Equipment I/O in progress (shared RAM or Parallel Port).
LED DS4 - SECS transmitter or receiver is transferring data.

If the PCC operating system encounters a serious problem during startup or during operation, this problem is indicated to the operator by flashing two or more LEDs. Errors are never indicated by one LED because this could be confused with the single blinking "operating" LED.

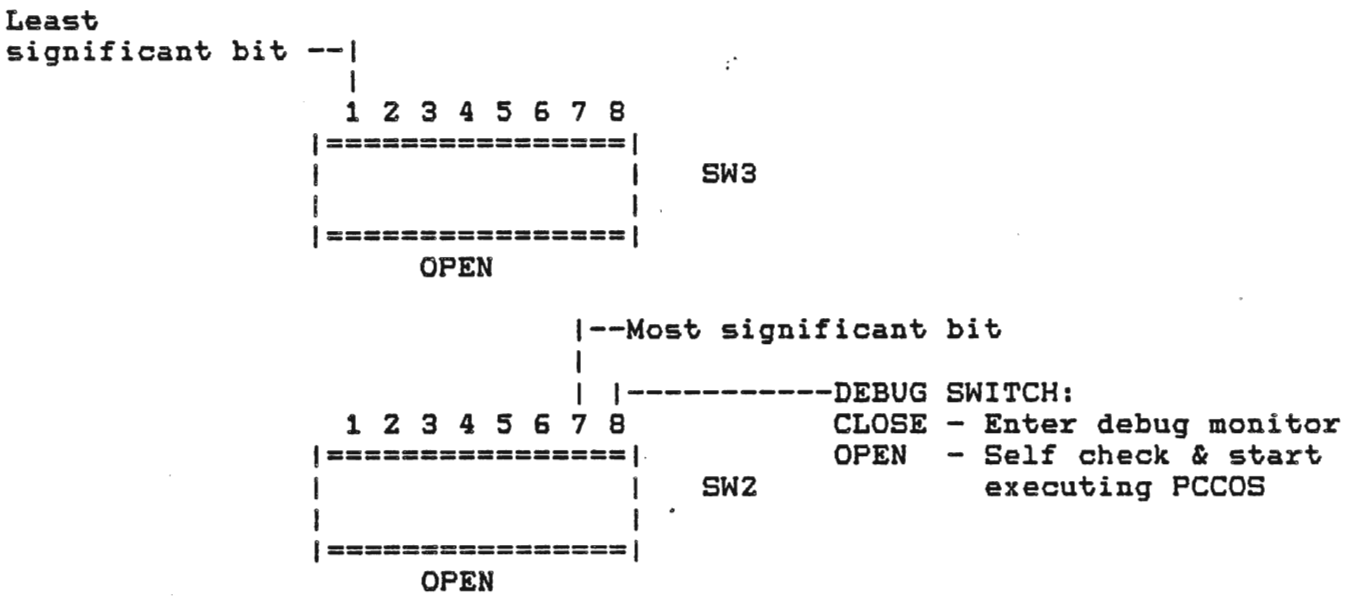
If two or more LEDs are flashing, the error can be determined from the list below. Note that DS1 is the least significant bit of the error code and DS4 is the most significant bit.

ERROR CODE	CONDITION
0011	-RAM check failed. #
0110	-ROM CRC check failed. *
1100	-E CLOCK signal from host processor not present

* - Location 0004 contains the expected CRC
  - Location 0006 contains the calculated CRC
- Location 0006 contains the faulty memory location.

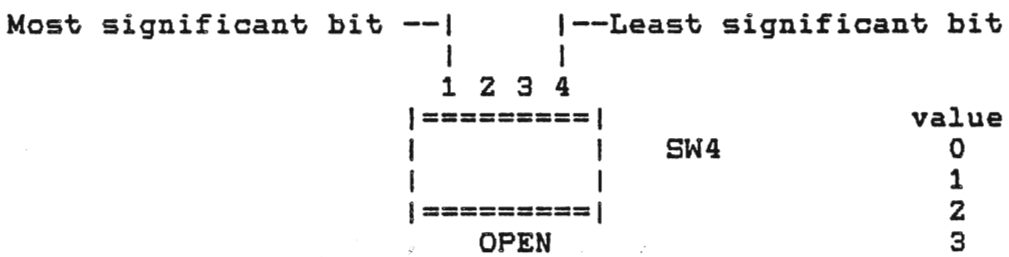
EXAMPLE: DS1 and DS2 on (code = 0011 - RAM check failed)

EQUIPMENT ADDRESS SELECTION



OPEN = 1
CLOSE = 0

SECS BAUD RATE SELECTION (REV B and later)



value	baud rate
0	*
1	19200
2	150
3	300
4	1200
5	2400
6	4800
7	9600

OPEN = 1
CLOSE = 0

* baud rate defined by personality module table

EXAMPLE: Switch 1 open; 2,3,4 closed = 0111 = 7 sets 9600 baud

APPENDIX H

PERSONALITY MODULE RESOURCES AND SELECTABLE PARAMETERS

The personality module has the following resources allocated for it's use:

1. 1024 bytes of Ram for local program variables (0600 - 0FFF)

2. 25 bytes of non-volatile Ram (addresses 4823 - 483C (16))

The baud rate of the SECS port and TT port are set by placing the appropriate value in the personality module entry & config table at offset PE.SBR or PE.TBR using the following values. Split rates are possible, the receive speed is defined by the most significant 4 bits, the transmit speed is defined by the low order 4 bits.
In PCC Rev B boards and later, these values
are used to set the SECS port baud rate only if the switches of SW4 equal 0 (all closed).

BAUD	-	VALUE
300	-	44
1200	-	66
2400	-	88
4800	-	99
9600	-	BB

APPENDIX I

PCC HARDWARE SPECIFICATION

KEN PAIST

E5100 Peripheral Communications Controller

1.0 Introduction

The Peripheral Communications Controller (PCC) is a single 6809 based board designed to compatible with E500 bus products. It is intended to serve as the single Factory Automation communications controller for all K&S automatic machines which are capable of FA interconnection.

2.0 Distinctive Characteristics

* 6809 MPU (1 MHz)

* 5 RAM/ROM sockets for industry standard 28 pin memories

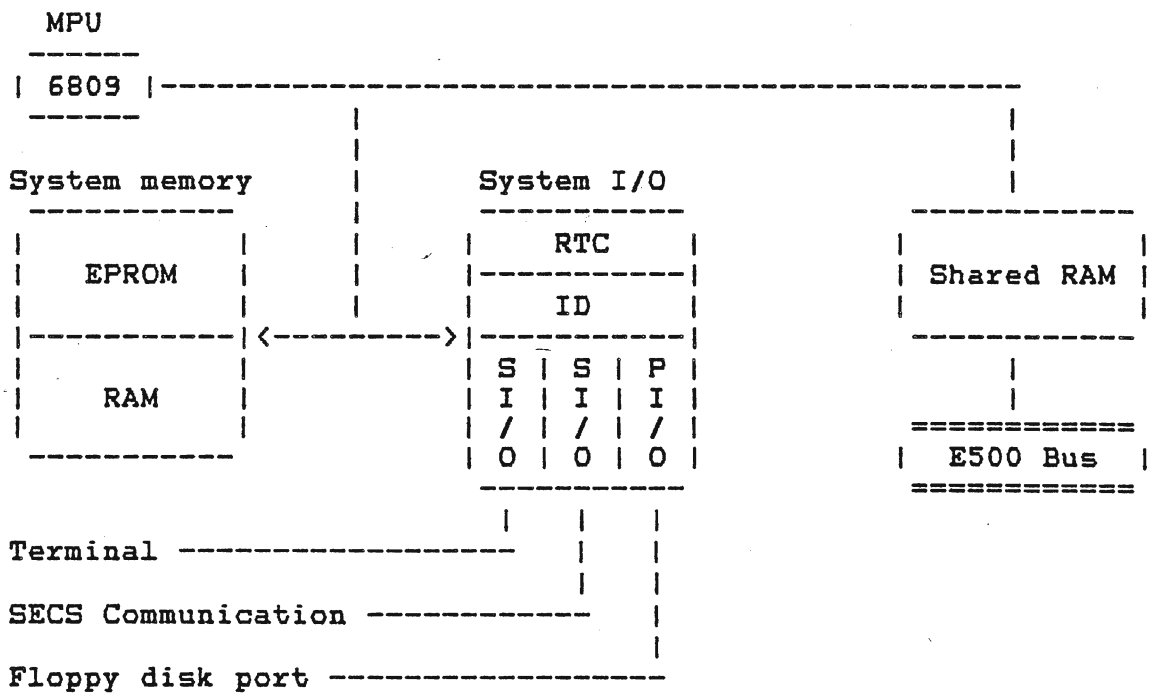
* 2 serial I/O with software baud rate select (110 to 38.4K baud)

* Real time clock/calender with 50 bytes of RAM and on board battery backup

* Shared memory interface with E500 bus products (including: 1482, 1471, 6300, 6100, 797, 380)

* Parallel interface via floppy disk port for non-E500 bus products (including: 1418/19, 1470, 775, 1500)

3.0 Block Diagram



4.0 Functional Description

The PCC is a single board communications controller designed to implement the SECS 1 communications standard for the majority of K&S automatic machines. It contains a 6809 MPU, RAM, EPROM, 2 serial I/O ports, an ID setting switch, a real time clock/callender, a parallel interface, a shared RAM interface and a third serial I/O interface for use by the automatic machine.

4.1 Processor

The processor is a 1MHz 6809 MPU. All three of its interupts are available to be jumpered to six interupting sources. The sources include 2 PIA interupts, 2 DUART interupts, a real time clock interupt and a shared RAM interupt. These interupts will be discussed in the sections describing the interupting source. A reset may be generated by an on board reset button, RESET* and POC* from the E500 bus. RESET* and POC* are jumper selectable. Both the DMA and HALT functions of the 6809 have been disabled.

4.2 System Memory

The system memory resides in five 28 pin RAM/EPROM sockets. These sockets are configurable to any combination of RAM and EPROM which conforms to the JEDEC approved 28 pin packaging. Address decoding is done with a 27S19 PROM so that the sockets may be enabled on 2K boundries. Pin 27, which may be WE* or PGM* for a RAM or EPROM respectively, may be jumpered to R/W or +5v. A typical configuration would include (3) 27128 EPROMs at the top of the memory space and (2) 6264-15 RAMs at the bottom. In this case, the lowest 4K block of the EPROM space would be disabled to allow for I/O and the shared RAM interface.

4.3 Serial I/O

The two serial I/O ports are contained on a 2681 DUART. It is accessed through 16 registers in the 6809's I/O space. It also includes an on-chip baud rate generator capable of producing rates from 50 baud to 38.4K baud under software control. The A port is setup to be the SECS communication port. It conforms to the SECS standard by providing plus and minus 12 volts to pins 18 and 25 respectively, transmitting on pin 2 and receiving on pin 3 of a female DB-25 connector. In addition to the standard, RTS and CTS are supported in hardware. The B port is configured as a DTE; pin 2 is transmit, and pin 3 is receive on a male DB-25 connector. The DUART also functions as an input and output port. As an output port, it generates the MSGRDY signal for the shared RAM interface. As an input port, it may be programmed to interupt the 6809 for servicing the shared RAM interface and/or sensing PFA* and/or LAES* on the E500 bus.

4.4 ID Field

The ID field is set by 16 bit dip switches that the 6809 can read as two bytes. This is included to conform to the SECS 1 standard. It's intended use is a machine ID.

4.5 Real Time Clock/calender

The real time clock/calender includes a time of day clock, a 100 year calender, interupting logic and 50 bytes of RAM. These functions are battery backed. Interupts may be programmed to occur periodically (.5 sec to 30.517 usec), as an alarm (once-per-second to once-per-day) and/or as an update-end. For more detailed information on these interupting modes, see the MC146818 data sheet.

4.6 Parallel Interface

The parallel interface consists of a PIA. It is used to interface the PCC with a 1418/19 or 1470 bonder through the bonder's floppy disk port. There are two interupt sources available to the 6809.

4.7 E500 Bus Interface

The E500 bus interface consists of a block of shared RAM and a serial I/O port. It is used to interface the PCC with any E500 bus machine.

Shared RAM is a high speed memory which is multiplexed between the machine's MPU and the PCC's 6809. It is designed to allow the machine's MPU to run full speed (no cycle stretching) at a 1MHz, 1.5MHz or 2MHz clock speed. The transfer and acknowledge byte is at the highest address of the shared memory, and the message ready bit is the most significant bit of that byte. It is active high (0 - no message 1 - message ready). When the machine writes to the MSB, the WRTOP signal becomes active. This can cause an interupt through the DUART to the 6809 indicating that a message is available from the machine. When the PCC has loaded a message into the shared RAM, it toggles a line which can cause an interupt to the machine (jumper selectable). The machine would sample the MSB of the shared RAM to test for the interupting source. If the interupt jumper is left out, the MSB would be polled for the message ready flag. The shared RAM may be configured as a 128 byte to a 1024 byte block of memory in the machine's memory space. There are some limitations to the memory size depending on the machine in which the PCC resides. The decoding logic can handle address bits A16 and A17 on machines which use them.

5.0 Implementations

There are two basic interface implementations on the PCC card, interface through the E500 bus and interface through a PAI. The E500 bus implementations communicate with the automatic machine through a shared memory block. Size and address of the shared memory is configurable for each machine. In the case of the PIA interface, a parallel data communications path is established through the bonder's floppy disk port. Since each machine has different hardware requirements, each machine will be personalized with PROMs and jumpers.

5.1 6300 Die Bonder

The 6300 Die Bonder interfaces with the PCC through the E500 bus. Its address bus is inverted, and therefore, it requires an inverting bus buffer to make address bits A0 to A7 non-inverted. The upper address lines are used for address decoding, so they need not be reinverted. The size of the shared memory may be 128 or 256 bytes. It may be located on any 128 or 256 byte boundry respectively. The host MPU clock speed may be between 1 and 2 MHz.

A line is available to interupt the host MPU. It may be connected to any of the interupt lines on the E500 bus or it may be left disconnected for a polled system.

The host's ACIA, which is fully decoded on the E500 bus, is designed to operate in 1MHz systems. It is not to be used in 2MHz systems.

5.2 1482 Wire Bonder

The 1482 Wire Bonder interfaces with the PCC through the E500 bus. The size of the shared memory may be 128, 256, 512 or 1024 bytes. It may be located anywhere in the bonder's memory space on a 128, 256, 512 or 1024 byte boundry respectively. Address bits A16 and A17 may be decoded with a host memory space greater than 64K.

A line is available to interupt the host MPU. It may be connected to any of the interupt lines on the E500 bus or it may be left disconnected for a polled system.

5.3 1471 Wire Bonder

Same as the 1482 (section 5.2)

5.4 6100 Die Bonder

Same as the 1482 (section 5.2)

5.5 797 Wafer Saw

Same as the 1482 (section 5.2)

5.6 380 Wafer Prep

Same as the 1482 (section 5.2)

5.7 1418/19 Wire Bonder

The 1418/18 Wire Bonder interfaces with the PCC through a parallel port. On the 1418/19 side the interface is the floppy disk port, while on the PCC side there is a PIA which is configured to look like a floppy disk controller. Messages are sent to and from the PCC the same way they are sent to and from a down load box. There are two interupt lines coming from the PIA which may be tied to any 6809 interupt line. It is important to note that the 1418/19 bonder is considered the master in all communications, and the PCC may not interupt the 1418/19. These constraints were imposed to relieve any real time requirements on the already interupt bound system.

5.8 1470 Wire Bonder

Same as the 1418/19 (section 5.7)

5.9 775 Wafer Saw

Same as the 1418/19 (section 5.7)

5.10 1500 HALT Machine

Same as the 1418/19 (section 5.7)

6.0 Environment

Temperature	0 to 50 degrees C
Humidity	up to 90% without condensation
Cooling	convection cooling is adequate, but forced air cooling is perfered

In general, there are no special requirements except that the equipment be operated in an invironment suitable for standard commercial grade electronic components.

7.0 Mechanical

as per K&S S103 standard size board

8.0 Power Requirements

+5v plus or minus 2%
ripple <+ .150v peak to peak
max. current =

+15v and -15v plus or minus 5%
ripple <= .5v peak to peak
max. current = 80mA (both plus and minus supplies)

9.2 Serial I/O

PCC "B" Connector Definition

B	Data	Signal name	B	Data	Signal name
B-1	DA-1	chasis ground	B-2	DA-14	
B-3	DA-2	TxD (SECS)	B-4	DA-15	
B-5	DA-3	RxD (SECS)	B-6	DA-16	
B-7	DA-4	RTS (SECS)	B-8	DA-17	
B-9	DA-5	CTS (SECS)	B-10	DA-18	+12v
B-11	DA-6		B-12	DA-19	
B-13	DA-7	GND (SECS)	B-14	DA-20	DTR (SECS) +12v
B-15	DA-8		B-16	DA-21	
B-17	DA-9		B-18	DA-22	
B-19	DA-10		B-20	DA-23	
B-21	DA-11		B-22	DA-24	
B-23	DA-12		B-24	DA-25	-12v
B-25	DA-13		B-26	DB-1	chasis ground
B-27	DB-14		B-28	DB-2	TxD (TT port)
B-29	DB-15		B-30	DB-3	RxD (TT port)
B-31	DB-16		B-32	DB-4	
B-33	DB-17		B-34	DB-5	
B-35	DB-18		B-36	DB-6	
B-37	DB-19		B-38	DB-7	GND (TT port)
B-39	DB-20	+12v	B-40	DB-8	
B-41	DB-21		B-42	DB-9	
B-43	DB-22		B-44	DB-10	
B-45	DB-23		B-46	DB-11	
B-47	DB-24		B-48	DB-12	
B-49	DB-25	-12v	B-50	DB-13	+5v

9.2 Parallel I/O

The parallel interface is comes through the 50 pin connector on the component side of the board. Signals connect directly to the PIA.

Pin	Function	Pin	Function
C1	GND	C2	
C3	GND	C4	
C5	GND	C6	EQPRDY
C7	GND	C8	EQPSIG
C9	GND	C10	
C11	GND	C12	PCCSIG
C13	GND	C14	PCCRDY
C15	GND	C16	PCCINT
C17	GND	C18	
C19	GND	C20	FD7
C21	GND	C22	FD6
C23	GND	C24	FD5
C25	GND	C26	FD4
C27	GND	C28	FD3
C29	GND	C30	FD2
C31	GND	C32	FD1
C33	GND	C34	FD0
C35		C36	
C37		C38	
C39		C40	
C41		C42	
C43		C44	
C45		C46	
C47		C48	
C49		C50	

9.3 E500 Bus

Pin	Function	Pin	Function
A1	-15v	A2	-15v
A3		A4	
A5	+15v	A6	+15v
A7	DIGITAL GND	A8	DIGITAL GND
A9		A10	
A11	+5v	A12	+5v
A13	+5v	A14	+5v
A15	POC*	A16	PFA*
A17		A18	RESET*
A19	NMI*	A20	IRQ7*
A21	IRQ6*	A22	IRQ5*
A23	FIRQ*	A24	R/W
A25		A26	E
A27		A28	
A29		A30	
A31		A32	
A33		A34	
A35	VMA	A36	IRQ4*
A37	IRQ3*	A38	IRQ2*
A39	IRQ1*	A40	A0
A41	A1	A42	A2
A43	A3	A44	A4
A45	A5	A46	A6
A47	A7	A48	A8
A49	A9	A50	A10
A51	A11	A52	A12
A53	A13	A54	A14
A55	A15	A56	A16
A57	A17	A58	IRQ0*
A59		A60	D0
A61	D1	A62	D2
A63	D3	A64	D4
A65	D5	A66	D6
A67	D7	A68	
A69	LAES*	A70	
A71	DIGITAL GND	A72	DIGITAL GND

APPENDIX J

REVISION HISTORY

Changes from previous release

13 Sep 84

PCCOS CHANGES

1. The shared RAM timeout semaphore is now defined to indicate that a message was not received by the PCC within the SECS T3 time. See shared RAM driver section for details.

2. At powerup, and after communication with the host fails, the PCC automatically polls the host with a "communication restored" stream 5, function 1) message until communication is restored.

3. LED definations have been redefined. Appendix G describes the meaning of each LED during operation and error conditions.

4. PCCs using a shared RAM interface now insure that the equipment E clock is present before accessing shared RAM.

5. When a PCC auto-starts (switch 8 of SW2 open) all debug messages are disabled. Use CNTL-C and then the D command to turn on debug message display.

SECS CHANGES

1. The SECS module automatically notifies the host if an invalid stream is received (stream 9, function 3).

2. The equipment ID of each message received from the host is checked to insure it was indeed intended for the equipment. A stream 9, function 1 message is sent to the host if the message was not for this equipment.

3. The SECS module notifies the host (stream 9, function 13) if a reply to a primary message is not received within the SECS T3 time. The personality module is still repsponsible for notifying the equipment.

4. Multi-block SECS messages are now checked for duplicates. Any duplicate block received by the PCC is discarded.